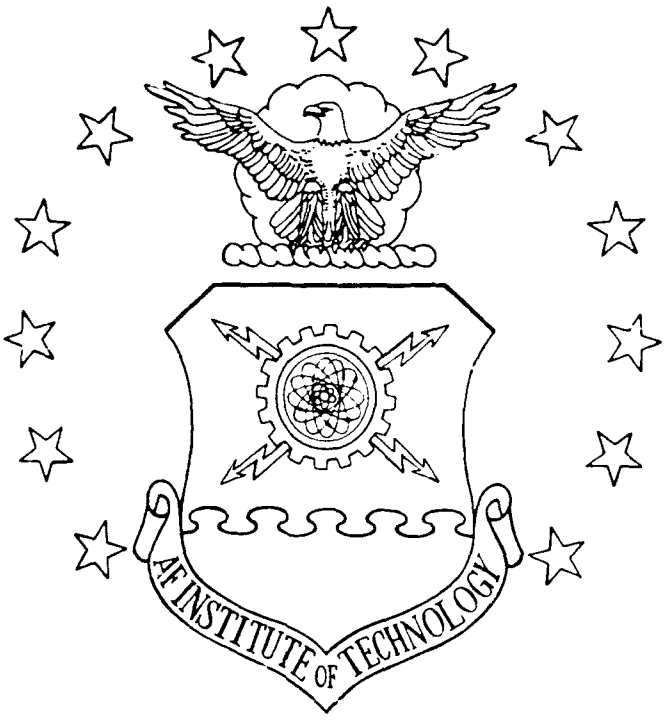

AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

DISSERTATION

Kerry D. Hicks
Captain, USAF

AFIT/DS/AA/89-1

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/AA/89-1

AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

DISSERTATION

Kerry D. Hicks
Captain, USAF

AFIT/DS/AA/89-1

AFIT/DS/AA/89-1

AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

DISSERTATION

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Kerry D. Hicks, B.S., M.S.

Captain, USAF

December 1989

AFIT/DS/AA/89-1

AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

Kerry D. Hicks, B.S., M.S.

Captain, USAF

Approved:

William E. Wiesel Jr. Nov 17, 1989

William E. Wiesel, Jr., Chairman

Rodney D. Bain Nov 17, 1989

Rodney D. Bain, Captain, USAF

Peter S. Maybeck Nov 17, 1989

Peter S. Maybeck

Vadim Komkov Nov 1, 1989.

Vadim Komkov

Accepted:

Robert A Calico Jr 17 NOV 1989

Robert A. Calico, Jr.
Interim Dean, School of Engineering

Acknowledgments

For almost three years now, I have worked on this research and finally, at long last, the finished product is ready for publication. To produce this, I had to rely on help and input from others. In particular, I'd like to thank my adviser, Dr. William Wiesel, for his guidance and enthusiasm in this subject area. I'd also like to thank Captain Rodney Bain for supplying the copy of ASAP used as the truth model, Ms. Billie Deason of the Johnson Space Center for providing technical data on Space Station Freedom, and Ms. Kristin Larsen for helping me endure (and recover from) the constant hardware failures in AFIT's computer systems. Thanks also goes to Nick Yardich and Jay Anderson for helping me obtain certain indispensable computer hardware.

In addition to technical support, I received my share of moral/sanity support for which I'm grateful: from my parents, my friends, and the entire staff of *Wiley's Comedy Club*. But, most of all, I'd like to thank my wife, Juanita. She deserves a medal for enduring the last five months as both a newlywed and an AFIT widow! Through it all, she has remained a source of patience, encouragement, love, and friendship when I needed it.

Thanks to all of you.

Kerry D. Hicks

Table of Contents

List of Figures

List of Tables

Notation

Listed below are the principal symbols used in this dissertation. All terms are scalars unless stated otherwise.

a	semimajor axis of orbit (3^{rd} state)
$a_f = e\cos(\Omega + \omega)$	equinoctial element (1^{st} state)
$a_g = e\sin(\Omega + \omega)$	equinoctial element (2^{nd} state)
CQ_nC^T	pseudonoise matrix (n x n)
D	diagonal matrix (n x n) (a factor of covariance matrix)
e	orbital eccentricity
$\hat{e}_s$	unit vector in direction of observed star
$\hat{e}_x, \hat{e}_y, \hat{e}_z$	unit vectors defining geocentric-equatorial system
$\underset{\sim}{f}$	set of dynamics equations (n x 1)
g_m, g_Ω, g_ω	orbit-dependent constants from Aksnes' theory
H	linearized formulation of observation relation (1 x n)
$h = \cos\gamma$	observation relation
i	orbital inclination
J_i	i^{th} zonal harmonic of Earth's gravity field
K	Kalman gain (n x 1)
$L = \Omega + \omega + M$	mean longitude of orbit (4^{th} state)
M	mean anomaly

n	mean motion
$\bar{n}$	"mean" mean motion
$\dot{\bar{n}}$	acceleration in mean longitude/mean anomaly parameter")
P	covariance matrix (n x n)
Q	variance of noise in observation measurement
t	time
U	unit upper triangular matrix (n x n) (a factor of covariance matrix)
v	zero-mean, discrete-time white Gaussian noise in observation measurement
$\overline{v_b^2}$	orbit-averaged speed squared of spacecraft relative to atmosphere
$\underset{\sim}{x}$	state vector in equinoctial elements (n x 1)
$\hat{\underset{\sim}{x}}$	estimate of state (n x 1)
$\underset{\sim}{x}_{ref}$	state about which system is linearized (n x 1)
$\underset{\sim}{x}_{g-e}$	state in geocentric-equatorial position and velocity (n x 1)
$z = \cos\hat{\gamma}$	measurement
$\Gamma = -\frac{2}{3\sqrt{\mu}} a_o^{5/2}$	shorthand notation
γ	true data (compliment of Earth-spacecraft-star angle)
$\hat{\gamma}$	measured angular data
γ_{bias}	bias in measured angle
δL_o	correction to mean longitude at epoch (one of two "epoch correction terms)

δn_o	linear correction term at epoch (one of two "epoch correction terms)
θ_{target}	angle between orbit plane and target star
$\pi = \Omega + \omega$	longitude of perigee
ρ	atmospheric density
σ_{comp}	standard deviation of star sensor/ Earth sensor combination
σ_{ii}	standard deviation of i^{th} state element
σ_θ	standard deviation of Earth horizon sensor
σ_ψ	standard deviation of star sensor
$\chi = \frac{\sin i \sin\Omega}{1 + \cos i}$	equinoctial element (5^{th} state)
$\psi = \frac{\sin i \cos\Omega}{1 + \cos i}$	equinoctial element (6^{th} state)
Ω	longitude of ascending node
ω	argument of perigee
$\dot{P}$	orbital period
P_o	rate of change of orbital period at epoch

Subscripts

o	evaluation at epoch
AK	computed with Aksnes' theory
RW	"real world" values
ave	average value

Superscripts

^{T}	transpose of matrix/vector
$^-$	pre-update value
$^+$	post-update value

Abstract

A new system for autonomous satellite navigation is developed and investigated. Unlike many previous studies, however, this system is not limited to the determination of an Earth satellite's current position. By using a two-step technique combined with a general perturbations model, this method allows the air-drag effect on the orbit to be estimated and applied to a future position prediction. Simulations using existing hardware have demonstrated that the algorithm presented is capable of current position estimates of sub-kilometer accuracy. Prediction precisions rival those of ground-based facilities: 8 - 12 km two weeks in the future for low-Earth orbits.

AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

I. Introduction

Almost all current United States Air Force space systems rely on ground-based control centers. This dependence is undesirable from two aspects. First, these centers are vulnerable to attack. Second, ground-based operations are rapidly becoming a major, if not *the* major, portion of a spacecraft's total mission cost (20:131). It is for these reasons that the U.S. Air Force/Space Division spacecraft autonomy program was initiated in 1980.

Two operations which presently require large amounts of ground-based processing time are orbit determination and data annotation. At present, orbit determination involves reducing data from worldwide spacetrack sensors to produce orbital element sets for objects in space. Data annotation is the combining of latitude, longitude, and time information with sensor data (such as photographs). Currently, these operations require an awkward and time-consuming mix of men and computers (30:1).

Responsibility for obtaining and maintaining current orbital element estimates for all Earth-orbiting objects currently falls on the Space Defense Center (SDC), located in the NORAD Cheyenne Mountain Complex (NCMC). With the number of satellites approaching 6000, it is

clearly desirable to reduce the computational load at SDC as much as possible (11:72). One possible way to achieve this reduction would be for individual satellites to share the burden with SDC by autonomously obtaining and maintaining estimates of their own orbital elements.

Orbit determination is but one function of an autonomous navigation system. To be truly useful, an autonomous navigation system should be capable of performing all of the other navigation functions currently executed on the ground. These include predicting future positions and events (such as eclipses) and performing correction maneuvers (38:III-66). At the heart of all of these functions is the necessity to know the position and velocity of the spacecraft in real time as well as the knowledge of how these will change with time. Thus, the goal of this research has been to find a system configuration and an algorithm to meet these requirements efficiently and accurately.

Chapter II reviews past research into this area. Chapter III provides an overview of the study and the simulation methods employed. Chapter IV derives and refines the short-term estimator used to calculate the spacecraft's current position, with the results being presented in Chapters V and IV. The problem of predicting positions far in the future is introduced in Chapter VII and an approach making use of a two-step process is derived therein. Chapter VIII contains the corresponding results.

Before detailing the methods derived and investigated in this research effort, it is important to outline past and present approaches to the autonomous satellite navigation problem. These will be discussed in the next chapter.

II. Methods and Related Efforts

Autonomous navigation systems can be separated into three distinct categories, organized by the type of data upon which they base their estimates (30:1). The groupings are:

1. Earth-based reference measurements.
2. Range measurements to known beacons in space.
3. Position sensitive angular measurements to celestial objects.

To understand the differences in these categories, they will be covered individually in some detail.

Earth-Based Reference Measurement Systems.

It is possible for space-based sensors to detect and track known landmarks on Earth. These landmarks may be natural features such as coastlines and lakes or may be man-made reference points such as highway crossings, airport radar emissions, and search lamps. Several technologies exist for this task: known landmark tracker/area correlations, known landmark trackers, artificial landmark trackers, and interferometer landmark trackers (30:5-8).

Known Landmark Tracker/Area Correlation. The theory behind known landmark tracker/area correlation is simple: known features in the field-of-view (FOV) of the sensor are used to enhance the determination of the satellite's location in space. While not an autonomous system by itself, the utility in this method is in its ability to enhance position

information from the navigation system significantly. When applied to LANDSAT data, this technique pinpoints the FOV to within 8 - 80 m (30:5-6).

Known Landmark Trackers. These systems utilize natural features, usually linear in appearance, such as rivers and coastlines. The concept involved is to detect and track the sharp discontinuities in intensity between a known feature and the surrounding area. Landmarks parallel to the flight path give cross-track information while those perpendicular yield in-track information (30:6-7).

Artificial Landmark Trackers. Identical in theory to the known landmark trackers just described, these differ only in that the feature tracked is man-made. Due to their easily recognizable characteristics, point references such as xenon search lamps may be used. Highways serve as excellent linear features (30:7-8).

Interferometer Landmark Trackers. Many radar units are dispersed about the globe at well-known locations. Passive use can be made of emissions from these sites. Two orthogonal phase interferometers measure phase differences in the received radar signal. This information yields a line-of-sight vector towards the radar. Knowing the radar's fixed position, the satellite's attitude, and this vector, the position of the satellite may be deduced. This is really just a special case of an artificial landmark tracking system (30:8).

All of these landmark techniques are ideal for image registration since they precisely fix the ground position being observed. Unfortunately, landmark observations cannot be used to determine the position

in space without knowing the attitude (30:5-8,10). Thus, these techniques cannot be used independently of an attitude determination system. Further, since these systems rely on Earth-based features, they must be considered vulnerable to man-made interference (jamming) as well as natural phenomena (clouds). Lowrie gives an excellent table of the benefits and limitations of each landmark system discussed (30:9).

Range Measurements to Known Beacons in Space.

Measurements made through observations of other spacecraft can be used by a navigation system to estimate the ephemeris of a satellite. These beacon satellites can, in theory, be actively participating in the tracking or be passive bystanders. But, for practical purposes, a high degree of accuracy is dependent upon beacons which actively participate by broadcasting, sharing information, etc.

Anthony's Proposal. Anthony has suggested a method for using relative motion data between two satellites to determine the orbital elements of each (5). His method relies on the ability to detect and track a nearby satellite for which an initial estimate of the orbital elements is known. While this could prove feasible between a rather large group of "friendly" satellites which crosslink information, it can hardly be considered completely autonomous. Further, since each satellite relies on others, the system is too vulnerable to attack.

Michael Ward's Proposal. Michael Ward's Master's thesis centered around satellite clusters (43). Using a simplistic model, Ward's research was a proof-of-concept study into a satellite's ability to determine its precise position relative to other nearby satellites. Each satellite in the cluster was assumed to be in a nearly circular

orbit and was able to obtain a measurement of the range to the others. As with Anthony's proposal, this means each satellite cannot be considered truly autonomous. A more direct impact of Ward's study to this research is his comparison of estimators applied to the problem. He showed, once again, the superiority of the U-D factorization of the Kalman filter (32:392-399) over the standard form of the Kalman filter (32:206-226; 43:2-14, 4-1).

GPS. The Global Positioning System (GPS) is another variation on the beacon idea. When completed, GPS is to consist of a constellation of 18 satellites, each broadcasting extremely accurate time and position information. By receiving broadcasts from all of the GPS satellites in view at any given time, a user can determine his position to within 5 - 70 m (1σ) and his velocity to approximately 0.002 m/sec (1σ). The actual precision is a function of the orbital altitude (14:34-35; 22:601).

While this system provides accurate position information, it is not without its drawbacks. First, it must be considered vulnerable; destroying a significant portion of the GPS constellation would render the system useless. In addition, the GPS satellites themselves are highly dependent on ground resources for their position and velocity information. Users of the GPS system must, therefore, be considered just as vulnerable as the ground centers themselves. Second, the GPS system does not provide any attitude information; therefore, any satellite using the GPS system would still require an independent attitude determination subsystem (30:11).

Position Sensitive Angular Measurements to Celestial Objects.

This area seems to be the most promising and best researched (14:27-32; 17; 21; 29; 30:1-3,4-5,10; 34; 35; 42). While many varied approaches exist, they share a common principle: by observing celestial bodies with known locations, a satellite's position in three-dimensional space can be deduced. Each approach to this method of position determination has its own unique characteristics, so this technique will be covered by example.

Lincoln Experimental Satellite Autonomous Stationkeeping System. In 1975, two satellites, LES 8 and 9, were launched by the U.S. Air Force to test the feasibility of autonomous navigation (30:2). The primary objective of these craft was to maintain the desired station as accurately as possible with a high fuel efficiency. Two sun transit sensors, two horizon sensors, and hardwired algorithms were used to determine the satellites' longitude and two degrees of attitude (roll and pitch). While this system was rather specialized and its accuracy modest, it demonstrated the feasibility of autonomous navigation (30:2).

AGN. The Jet Propulsion Laboratory's Autonomous Guidance and Navigation (AGN) project was designed for use in interplanetary missions (30:2). This system uses solid state sensors to detect stars, planets, asteroids, and other illuminated bodies. The absolute directions of background stars are determined and compared to an onboard star catalog for identification. Once stars are identified, the measurements can be used to determine the satellite's attitude and position. Further, with additional knowledge about the relative location of an illuminated target (such as a planet), the system can deduce the target's ephemeris.

Space Sextant. The space sextant autonomous navigation and attitude reference system (SS/ANARS) is being built by Martin Marietta Aerospace for the USAF (14:30-32; 17:7-8,16-17; 30:4-5). The sextant is composed of two cassegranian telescopes, accurate to 0.5 arcseconds, mounted on a three-degree-of-freedom platform. One telescope tracks the bright limb of the moon, while the other tracks a known star. Navigation information is derived from the included angle between the two lines of sight. The system is advertised to yield the attitude to 0.6 arcsec. (1σ), position to 244 - 300 m (1σ), and velocity to 0.03 - 0.003 m/s (1σ). Further, the expected flight mass and power consumption are 25 kg and 50 W, respectively (14:30,32; 17:16; 30:9).

MADAN. TRW's Multimission Attitude Determination and Autonomous Navigation System (MADAN) incorporates two star sensors, an Earth sensor, and a computer (14:28,30; 17:7). Angular data from the star sensors is sufficient to derive the spacecraft inertial attitude. Knowledge of the attitude, when combined with the measurements from the Earth sensor, allows the position of the satellite with respect to the Earth to be computed. Navigational precision is said to be 0.4 - 1.4 km (1σ) in position and 0.4 - .13 m/s (1σ) in velocity for orbital altitudes less than or equal to geosynchronous altitude. Mass and power consumption characteristics are comparable to those of the space sextant (14:28,30; 17:7).

SHAR/SHAD. Stellar Horizon Atmospheric Refraction (SHAR) (21) and Stellar Horizon Atmospheric Dispersion (SHAD) (29) techniques make use of the refractive/dispersive properties of the Earth's atmosphere. As the line of sight between a satellite and a star encounters the Earth's

atmosphere, it is refracted and dispersed. If the relationships between altitude and these variations are known, then one component of the satellite position can be determined with each observation. Multiple observations yield a complete set of orbital elements. With extremely accurate sensors (thousandths of arcseconds), the standard deviation in position determination is advertised to be 100 - 250 m, depending on orbital altitude (14:35; 21:129,133; 29:573-574,583-587).

Mease et al.'s Proposal. Mease et al. have proposed a method of autonomous navigation for geosynchronous satellites using instruments already on many such spacecraft (34). A sun sensor, an Earth sensor, and a solar-array-drive potentiometer supply the data. An extended Kalman filter produces an estimate of the spacecraft's orbital elements from the observations.

Although somewhat limited in application by the requirement the satellite be in synchronous orbit, this method does show promise. Within the first 24 - 36 hours, the filter converges to the vicinity of the true state with an error standard deviation of 2 km in the osculating semimajor axis and 0.03° in the mean inertial longitude (34). Statistics on other errors such as that in the inclination angle were not available.

Metzler's Proposal. Metzler has studied the ability of an autonomous system for satellites operating in high-altitude orbits (35). His study investigates the possibility of using a star sensor, an Earth sensor, and a linear Kalman filter to estimate the position and velocity of a satellite in a circular orbit at five times synchronous altitude. The steady-state error in estimated position for this

system is predicted to be about 12 km (1σ). Predicted steady-state velocity errors are on the order of 0.15 m/s (1σ) (35).

John Ward's Proposal. In John Ward's Master's thesis, he proposed using one or two star sensors and an Earth sensor to determine the orbital elements of a satellite in an arbitrary orbit (42). Using two-body equations of motion and simulated data, he used both nonlinear least squares and Bayes filter methods (42:27-33; 44:64-69,91-93) to produce position estimates. Since Ward's method involved numerical integration, its applicability to current, space-qualified onboard computers is questionable. This is because onboard computers are, typically, small and numerical integration of the equations of motion and the equations of variation can be a very resource-demanding process (44:77). Ward's study did, however, demonstrate that a configuration combining star sensors and an Earth sensor has the capability of being used to produce position estimates accurate to at least two or three kilometers (42:63).

Conclusion

As is clear from the long list of related efforts, autonomous navigation is a much desired goal. Each of the described methods and studies has its advantages and disadvantages. One common disadvantage that all of these methods share is the inability to predict accurately far into the future -- the so-called, "Where will I be?" question. In part, this is due to the inability to estimate air-drag parameters with much accuracy. Further, the simple dynamics models used in many of these systems ignore drag effects altogether, leading to large errors when one attempts to propagate the trajectory into the future. This

problem will be addressed in Chapter VII, "The Formulation of the Long-Term Estimation Problem" and the method derived will be studied in a simulated application to NASA's Space Station.

Perhaps the largest drawback of the systems intended for Earth-orbit applications is that none are capable of using inexpensive, "off-the-shelf" hardware to produce accurate estimates of current and future positions for satellites in arbitrary orbits. In fact, not a single truly autonomous system for Earth satellites is in use today (45). One major contribution of this research is the development of a system which relies only on sensors and microprocessors in use today.

III. Research Overview

The purpose of this research has been to develop and test an algorithm for autonomous satellite navigation relying only on currently available hardware. Unlike the schemes discussed in the previous section, this system has been designed with more in mind than just the desire to estimate a satellite's current position. The procedure developed herein not only finds the satellite's current orbital position, but can also predict how this changes over long periods of time. Thus, this system is able to perform both short-term ("Where am I?") and long-term ("Where will I be?") estimation.

Estimator

The problem examined is the derivation of a satellite's position from imperfect observations and measurements. Such problems are precisely the function of estimators/filters. This study applies an iterated, extended form (33:58-59) of the Bierman-Thornton U-D filter (9:77,95-100,149; 32:392-397). The U-D factorization has well-documented superiority over other sequential estimators in applications such as this one (9:77,95-100,149; 13:1; 16:444; 17; 24:223; 32:236,399; 39:1-3; 43; 44:104-105).

Sensor Configuration

The satellite configuration considered was one incorporating at least one star sensor, an Earth sensor, and a microcomputer. This arrangement of sensors was selected because it is a common combination on existing satellites (42:14). Metzler and J. Ward have both shown

that accurate position determination is possible with such instrumentation (35; 42).

J. Ward found that using data from two star sensors produced better results than using just one (42:55,56); however, as will be shown in a later section, the generalization that "more is better" does not always apply to the system developed here. Indeed, it was found that the proper choice of a single observation star was typically far better than the haphazard choice of two. Thus, a part of this investigation was into the effect of star selection; for instance, how important is the location of the viewed star relative to the satellite's orbit?

Dynamics Model

The estimator requires a dynamics model to propagate the state estimate forward from one observation time to the next. Solving the long-term estimation problem also requires equations of motion to propagate the states forward for long periods; therefore, a model of the dynamics must be included in the navigation system.

This is a real-time application on a microcomputer, so numerical integration is best avoided if alternative methods can produce comparable precision with fewer computations. General perturbations solutions (closed-form) can deliver such performance. Further, autonomous navigation studies involving numerical integration are common in the open literature while studies involving general perturbations methods are not (10; 11; 17; 22; 39; 42). For these two reasons, a general perturbations theory derived by Aksnes' was selected as a starting point (2; 3; 4).

Air drag significantly affects satellite motion in low-altitude orbits. Predicting the position of a satellite far into the future requires some knowledge of this effect. Other attempts have involved modifying the equations of motion (10:13-14; 11:9-11; 25). For example, Boden approached this problem by estimating the ballistic coefficient of the satellite concurrently with the orbital element estimation (11:1, 9-11). His attempt failed to produce an accurate or computationally practical method for onboard orbit prediction.

A fundamental problem with Boden's and similar research is that the effect of air drag between observations is so small as to be overshadowed by the error in sensor measurements (44:95-97). In addition, previous methods have tended to account for drag by working its effect into the semimajor axis and eccentricity calculations only (11.9-11; 25:6). Jazwinski suggests the effect of air drag is better observed as an effect on the mean anomaly (23:306). Experience in orbit prediction at the NCMC further supports this suggestion (10:13; 45). Indeed, an analytical basis for this effect can be derived and is presented in an upcoming section.

This research investigates tracking the drag effect as a two-step estimation problem. First, observations are taken at a rate for which drag effects cannot be separated from random noise in the sensors. This supplies enough data to produce accurate estimates for the orbital elements at any given time. (In other words, this data is meant to answer, "Where am I?") Then, after a sufficient time (approximately one week), the effect of air drag manifests itself as an increase in the mean anomaly that is quadratic with time. (A corresponding linear decrease

in the semimajor axis can also be observed, only to a lesser degree.) At that point, an estimate of the drag effect can be determined. This two-step estimation technique was investigated by simulating an application to a situation where drag is quite significant: NASA's Space Station Freedom (15).

Onboard Computer Simulation

The satellite's computer was assumed to be equivalent to an 8 MHz 8088-based microcomputer with an 8087-2 math coprocessor. This configuration at least approximates current space technology (45). In reality, a satellite might be time-sharing this computer between navigation, attitude determination, data annotation, health and maintenance functions, etc. This research looked into the tradeoff between computer time allotted to navigation and the precision to which the position can be estimated.

Data Simulation

To simulate a satellite in space taking measurements, a simple procedure was followed. As shown in Figure 3-1, perfect data was first generated using an accurate truth model [the Jet Propulsion Laboratory's Artificial Satellite Analysis Program (ASAP) (28)]. The true orbit was assumed to be adequately represented by a model consisting of an Earth with a 6 x 6 gravity field and an oblate, rotating, exponential atmosphere. The geopotential values employed are tabulated in Appendix A. Further, third-body effects were limited to those due to the Sun and moon. (For completeness, note the initial Earth, Sun, moon, and

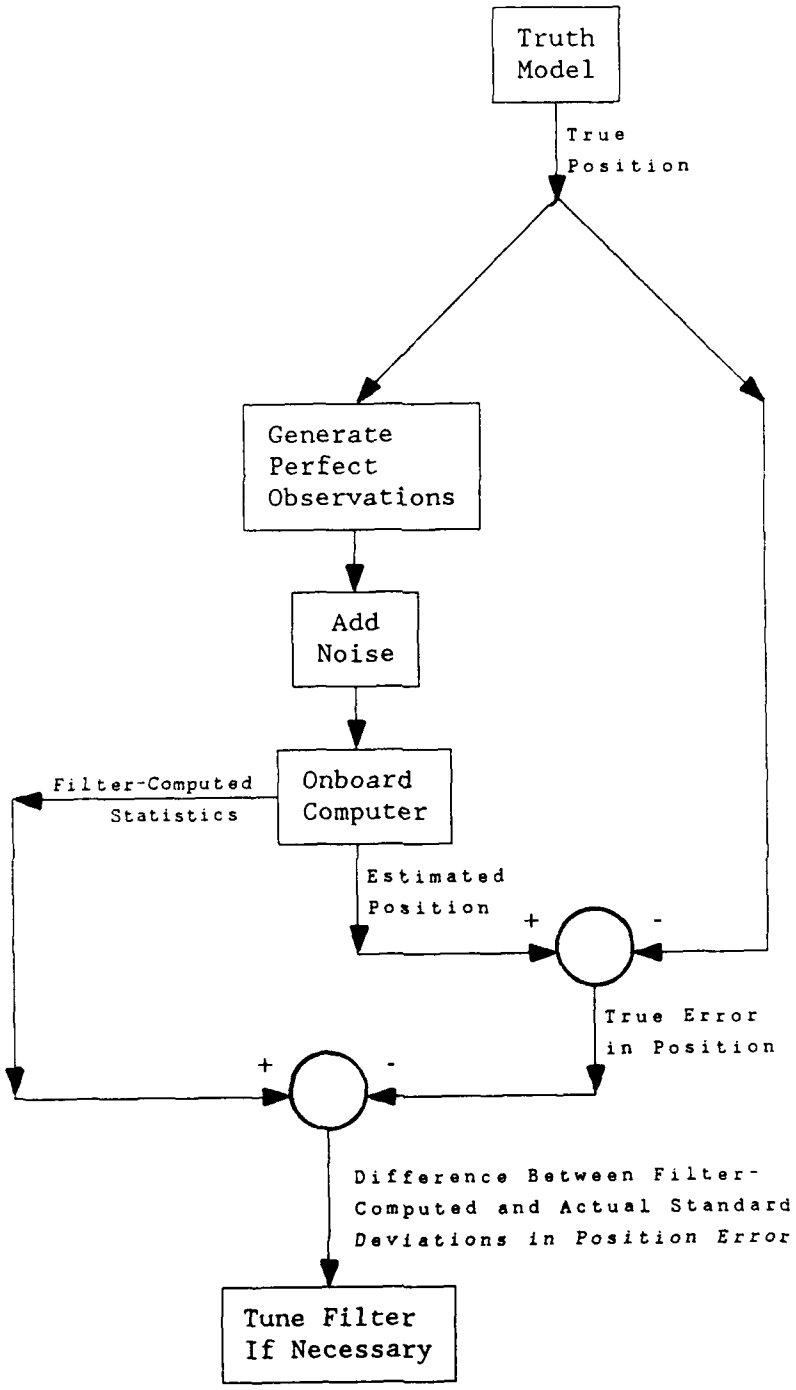

Figure 3-1. Data Simulation

satellite positions were arbitrarily tied to an epoch date of 1 January 1988.) Simulated data was generated from this true state model. Then to reflect "the real world" of imperfect instruments, zero-mean discrete-time white Gaussian noise was added to the data (32:9; 44:5). The "noisy" data was then fed into the microcomputer as if it were onboard receiving sensor data. (The effect of biased noise was investigated as a special case. Results are presented in Chapter V.)

The estimator's performance was evaluated by comparing the estimated position with the true position over an adequate number of Monte Carlo runs. To determine how many individual simulations were needed to be "adequate," the number of runs included in the Monte Carlo analysis was increased until consecutive results ceased to show any significant change. The true standard deviation in the error of the estimate was obtained and compared to the corresponding standard deviation estimate predicted by the filter, allowing the filter to be "tuned" to match predicted error with the true error. This comparison is also shown in Figure 3-1.

Summary Of Research

The primary goal of this research was the development of an algorithm to not only generate a satellite's position from sensor data, but to also predict future positions when necessary. This procedure was developed for a satellite consisting of star sensors, an Earth sensor, and a microcomputer. Onboard software includes a general perturbations model for the dynamics and a filter to produce position estimates from imperfect observations.

The uniqueness of this study lies in three aspects. First, only currently available technology is exploited. This approach insures minimum implementation delays and costs. Second, the two-step approach to estimating drag effects is not, apparently, used in orbit determination at the present time, particularly in an autonomous situation. Finally, the use of this general perturbations model to describe the dynamics in a space-based autonomous system is new. In particular, a modification of Aksnes' model to include drag effects is unique.

IV. The Formulation of the Short-Term Estimation Problem

The problem of a satellite autonomously determining its orbital position can be termed a short-term estimation problem since the times between estimates are short. Descriptively, this can be referred to as the "Where am I?" problem. All of the studies covered in Chapter II were capable of solving this problem. Each combined an estimator, some sort of dynamics model, and data observations to accomplish the task. In that respect, this study is no different. Rather, the uniqueness in this research is in the novel combination of software and hardware that allows an accurate system to be put into operation utilizing only currently available resources.

The Filter

Mathematically, the problem at hand is to estimate the state $\underset{\sim}{x}(t_i)$ of a system for which an approximate nonlinear dynamics model is known

$$\underset{\sim}{x}(t_i) = \underset{\sim}{f}\left[\underset{\sim}{x}(t_o), t_i\right] \tag{4-1}$$

where $\underset{\sim}{x}(t_o)$ is the state at the epoch time t_o. Note, $\underset{\sim}{f}$ is a closed-form (analytic) solution to the equations of motion and *not* a set of differential equations. Further, at time t_i a scalar observation $z(t_i)$ is made. This observation is related to $\underset{\sim}{x}(t_i)$ by the following nonlinear observation relation:

$$z(t_i) = h\left[\underset{\sim}{x}(t_i)\right] + v(t_i) \qquad (4\text{-}2)$$

$v(t_i)$ is zero-mean discrete-time white Gaussian noise, independent of $\underset{\sim}{x}(t_i)$, with a variance of Q. Recall, $\underset{\sim}{f}$ is only an approximate solution and not exact, so Eq. (4-1) contains errors that must be dealt with in some manner. This will be accomplished shortly by adding process noise of the form $Cw_n(\Delta t)$ when Eq. (4-1) is linearized.

All of the approaches previously described produce state estimates from imperfect data, with the state being the spacecraft's position and velocity or, equivalently, a set of orbital elements. For best results, the effect of noise in the data must be minimized in a statistical sense. Since this is precisely the function of a filter/estimator, navigation problems lead to the application of some sort of estimator (18:557; 32:4-5). For example, Mease, et al. (34) and Metzler (35) employed Kalman filters, while J. Ward used both least-squares and Bayes filters (42). These algorithms, and others like them, combine all available measurement data, plus prior knowledge of the system and the precision of the data, to produce an estimate of the state.

Filter Choice. As noted above, various estimation algorithms could be employed for this problem. These include the least squares, the Bayes, and the various forms of the Kalman filter. In this situation, however, the selection of which filter to employ is more a process of elimination than a choice. This is a real-time application, so the least squares and Bayes approaches are undesirable because they require a matrix inversion on the order of the state vector. This leaves the

various forms of the Kalman filter as they only involve a matrix inversion on the order of the data vector (a scalar in this application) (44:100-102).

Due to its numerical instability, the conventional Kalman filter should not be selected for use in small computers (9:95-100; 32:236; 39:1-3; 44:104-105). Factorization, or so-called "square root" and "U-D," formulations of the Kalman algorithm, have, however, demonstrated their stability and accuracy (9:77,149; 13:1; 16:444; 24:223; 39:2). According to Maybeck, square-root filters can yield twice the effective precision of conventional Kalman filters in poorly conditioned problems (32:369). U-D formulations also have this numerical robustness, but do not require the computation of numerous square roots; thus, they can be the faster than square-root filters (40:III-89).

In particular, the Bierman-Thornton U-D filter has been shown to be a numerically stable alternative to the Kalman formulation (32:399; 39:2). Its superiority for this type of problem has been demonstrated in other orbit-determination problems (17; 39; 43). Junkins goes so far as to say that the U-D filter is "superior to all known methods" when propagating the covariance matrix from one observation time to the next (24:223). This formulation involves factoring the covariance matrix into the form

$$P = UDU^T \tag{4-3}$$

where U is a unitary upper triangular matrix (unity on the diagonal of an upper triangular matrix) and D is a diagonal matrix. Thus, this

filter computes and propagates U and D (and, of course, a state estimate, $\hat{x}$) rather than P.

In spite of its apparent complexity, the computational burden of this filter is comparable to that of the Kalman filter. Numerical studies by Thornton and Bierman demonstrate this fact (39:25). Analytical operation counts further support this finding (9:78,84-90; 32:402-404). Even the storage requirements for this filter can be shown to be comparable to the Kalman algorithm if care is taken to exploit the special forms of U and D (40:III-89). While the problem at hand is nonlinear and the above observations based on linear formulations, one would expect these statements to hold true in general.

As just stated, the problem studied is nonlinear; in fact, it is *highly* nonlinear. Iterated, extended formulations are capable of providing better performance than their linear or extended counterparts in situations such as this (33:58-59). For this reason and all of those cited above, an iterated, extended form of the U-D filter was selected for use in this research. To help introduce the notation, the form of the U-D factorization filter employed in this research will now be derived.

Iterated, Extended U-D Filter Derivation. The U-D formulation is really just a mathematical rewriting of the Kalman algorithm. More precisely, since this is a nonlinear problem, the filter to be derived is a reformulation of the iterated, extended Kalman filter. Simply put, this is an iterated form of a linearized factorization filter. For the derivation, it is sufficient to start with the iterated, extended Kalman filter as cited by Maybeck (33:39-45,58-59).

First, some notation should be introduced:

(a) $\hat{\underset{\sim}{x}}(t_i^-)$ and $\hat{\underset{\sim}{x}}(t_i^+)$ are estimates of the state at time t_i before and after the the incorporation of the observation at time t_i, respectively.

(b) $P(t_i^-)$ and $P(t_i^+)$ represent the covariance matrix before and after incorporation of the observation, respectively.

(c) $\underset{\sim}{x}_{ref}(t_i)$ is the current reference trajectory about which linearization of the observation relation occurs.

While this notation is explicit, it is also a bit cumbersome. When no chance of confusion exists, it will be simplified. For example, $\hat{\underset{\sim}{x}}(t_i^-)$ will be written as $\hat{x}(-)$, $P(t_i^+)$ as $P(+)$, $K(t_i)$ as K, etc.

The linearization of Eqs. (4-1) and (4-2) is given by

$$x(t_{i+1}) = \Phi(t_{i+1},t_i)x(t_i) + Cw_n(\Delta t)$$

$$z = Hx(t_i) + v(t_i) \qquad (4\text{-}4a)$$

where:

$$\Phi(t_{i+1},t_i) = \left.\frac{\partial f}{\partial x_i}\right|_{x_{ref}} \qquad (4\text{-}4b)$$

$$H = \left.\frac{\partial h}{\partial x}\right|_{x_{ref}} \qquad (4\text{-}4c)$$

$w_n(\Delta t)$ is a discrete-time function that describes the error in the state estimates due to modeling errors if $\underset{\sim}{f}$ and the linearization (32:145-146). For the purposes of the filter, $w_n(\Delta t)$ can be assumed, for lack of any better assumption, to be zero-mean discrete-time white noise. Indeed, if the error is comprised of many small, random errors, then the Central Limit Theorem states that the function will approach Gaussian (32:109; 44:5,7). Any error in this assumption can be empirically accounted for when the filter is tuned. The matrix C describes how $w_n(\Delta t)$ affects each state. $\Phi(t_{i+1}, t_i)$ is well-known as the state transition matrix.

The Kalman gain is given by:

$$K = P(-)H^T\left[HP(-)H^T + Q\right]^{-1} \tag{4-5}$$

Note that, since the observation and, hence, Q, is a scalar quantity, the inversion in Eq. (4-5) is simply the reciprocal of a scalar.

In the iterated, extended Kalman filter, it is necessary to revise the reference state at each iteration. This revision is given by

$$x_{ref} = \hat{x}(-) + \delta x \tag{4-6a}$$

where

$$\delta x = K\left\{z - h - H\left[\hat{x}(-) - x_{ref}\right]\right\} \tag{4-6b}$$

with h and H evaluated using x_{ref}. A useful stopping criterion for

the iteration scheme is to repeat the procedure until every correction to the state is smaller than its associated uncertainty; i.e., $\delta x_i < \sqrt{P_{ii}(-)}$ (10:18; 42:101; 44:67).

The equations relating estimates before and after an observation are referred to as the update equations. These are

$$\hat{x}(+) = x_{ref} \tag{4-7}$$

$$P(+) = P(-) - KHP(-) \tag{4-8}$$

where x_{ref} is the reference state from the last pass of the iteration scheme and H is evaluated using $\hat{x}(+)$.

The last step needed to complete the procedure is a set of equations to propagate the state estimate and covariance forward from time t_i to time t_{i+1}. These are simply

$$\hat{\underset{\sim}{x}}(t_{i+1}^-) = \underset{\sim}{f}\left[\hat{\underset{\sim}{x}}(t_i^+), t_{i+1}\right] \tag{4-9}$$

$$P(t_{i+1}^-) = \Phi(t_{i+1}, t_i)P(t_i^+)\Phi^T(t_{i+1}, t_i) + CQ_n(\Delta t)C^T \tag{4-10}$$

where $\underset{\sim}{f}\left[\hat{\underset{\sim}{x}}(t_i^+), t_{i+1}\right]$ is a closed-form set of dynamics equations relating the states at t_i^+ and t_{i+1}^- and Q_n is the covariance of the dynamics driving noise w_n (32:220; 44:109). The product CQ_nC^T serves two purposes. First, it corrects the covariance matrix to account for uncertainties in the dynamics model over Δt. Thus, the filter-computed statistics can be made to match the true statistics. Second, it allows

the filter to be "tuned" to prevent the covariance matrix becoming singular -- a disastrous condition for sequential filters (33:23-28; 44:105-106).

To determine the elements of CQ_nC^T, Monte Carlo simulations can be performed. By running a sufficiently large number of simulations, the elements of this matrix product can be varied until the filter-computed covariance, P(+), is representative of the true statistical error in the estimate. In other words, diagonal elements of the covariance should at least approximate the true variances in the state estimates and off-diagonal terms should be representative of the true correlation between the states.

Eqs. (4-4) through (4-10) can be assembled to form the iterated, extended Kalman filter. This is best visualized when written as a computer-style flowchart (See Figure 4-1).

To convert this algorithm into a factorization algorithm, recall that any $n \times n$ symmetric, positive semidefinite matrix can be factored as

$$P = UDU^T \tag{4-11}$$

where U is an $n \times n$ unitary upper triangular matrix and D is an $n \times n$ diagonal matrix (32:392). Noting the covariance matrix meets these restrictions and observations are scalar quantities, the Kalman gain [(Eq. (4-5)] can be written

$$K = \frac{U(-)D(-)U^T(-)H^T}{a} \tag{4-12a}$$

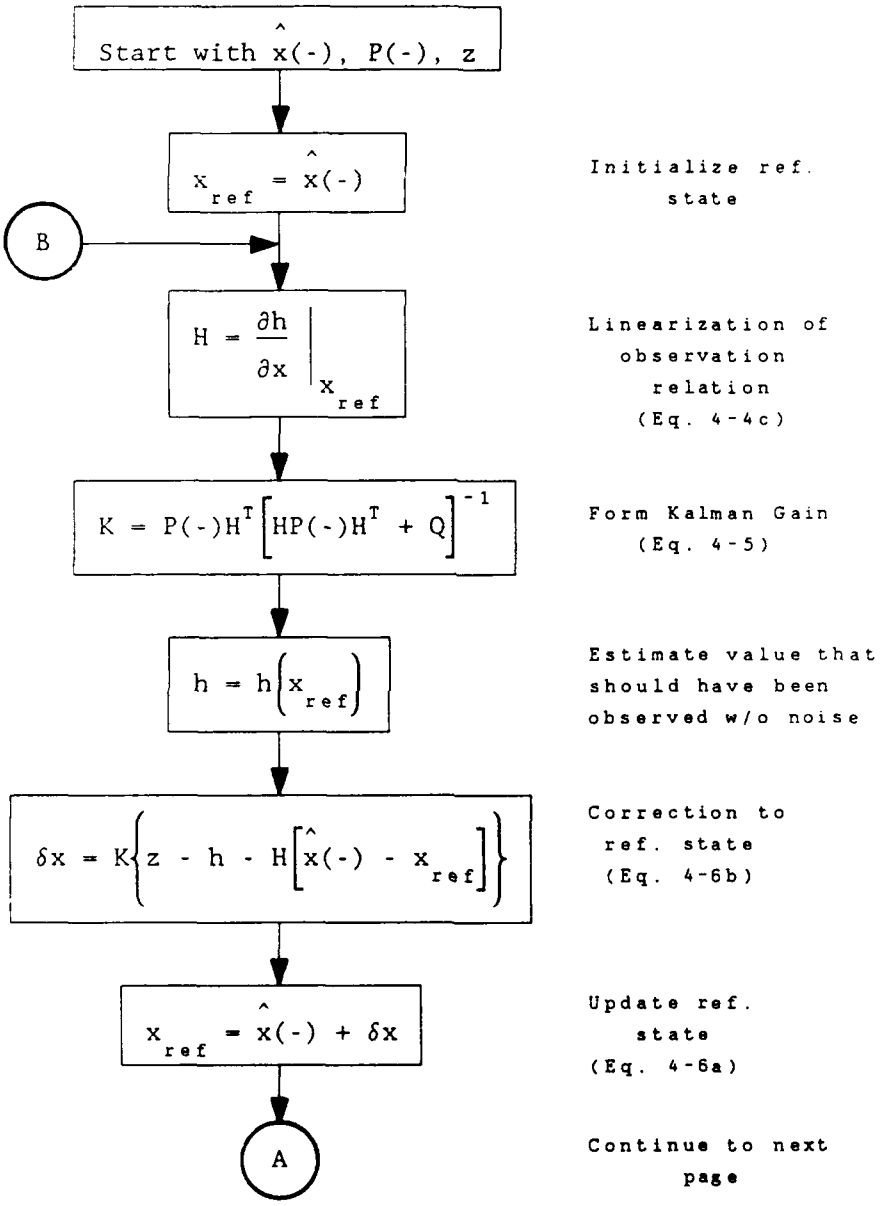

Figure 4-1. Algorithm for Iterated, Extended Kalman Filter (Part 1 of 2)

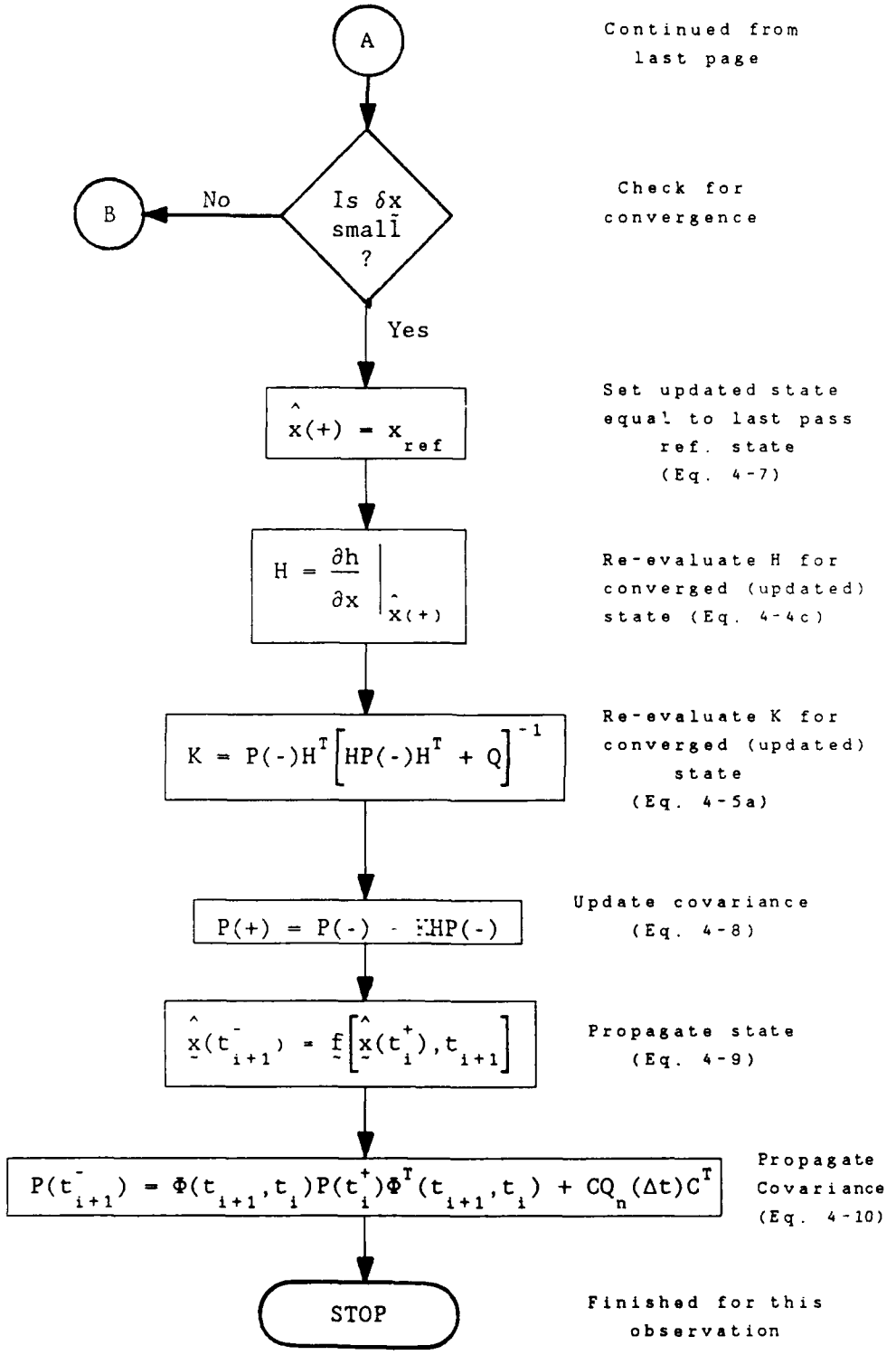

Figure 4-1. Algorithm for Iterated, Extended Kalman Filter
(Part 2 of 2)

where:

$$a = HU(-)D(-)U^T(-)H^T + Q \tag{4-12b}$$

Similarly, the covariance update equation [Eq. (4-8)] can be factored to give:

$$U(+)D(+)U^T(+) = U(-)D(-)U^T(-) - KHU(-)D(-)U^T(-) \tag{4-13}$$

Incorporating Eqs. (4-12) and factoring the right-hand-side, this becomes:

$$U(+)D(+)U^T(+) = U(-)\Big\{D(-) - \left(\frac{1}{a}\right)\Big[D(-)U^T(-)H^T\Big]\Big[HU(-)D(-)\Big]\Big\}\, U^T(-) \tag{4-14}$$

Since $D(-)$ is diagonal:

$$D(-) = D^T(-) \tag{4-15}$$

Thus, Eq. (4-14) can be rewritten:

$$U(+)D(+)U^T(+) = U(-)\Big\{D(-) - \left(\frac{1}{a}\right)\Big[D(-)U^T(-)H^T\Big]\Big[D(-)U^T(-)H^T\Big]^T\Big\}\, U^T(-) \tag{4-16}$$

If there are n state variables, then the n-vectors $\hat{\underset{\sim}{f}}$ and $\underset{\sim}{v}$ can be defined as:

$$\hat{\underset{\sim}{f}} = U^T(-)H^T \tag{4-17a}$$

$$\underset{\sim}{v} = D(-)\hat{\underset{\sim}{f}} \tag{4-17b}$$

Then, Eq. (4-16) can be simplified to read:

$$U(+)D(+)U^T(+) = U(-)\left[D(-) - \left(\frac{1}{a}\right)\underset{\sim}{v}\underset{\sim}{v}^T\right]U^T(-) \tag{4-18}$$

If the bracketed term is a positive, semidefinite, symmetric matrix, it could be factored as:

$$\left[D(-) - \left(\frac{1}{a}\right)\underset{\sim}{v}\underset{\sim}{v}^T\right] = \bar{U}\bar{D}\bar{U}^T \tag{4-19}$$

Symmetry is apparent, but the proof of positive, semidefiniteness for the general case is non-trivial; however, for the scalar case it is simple enough to warrant inclusion here. For scalars:

$$a = H^2U^2(-)D(-) + Q \tag{4-20a}$$

$$\hat{f} = U(-)H \tag{4-20b}$$

$$v = D(-)\hat{f} = D(-)U(-)H \tag{4-20c}$$

Therefore,

$$\left[D(-) - \left(\frac{1}{a}\right)\underset{\sim}{v}\underset{\sim}{v}^T\right] = D(-) - \frac{D^2(-)U^2(-)H^2}{H^2U^2(-)D(-) + Q} \tag{4-21}$$

$$= D(-)\left[1 - \frac{1}{1 + \frac{Q}{H^2U^2(-)D(-)}}\right]$$

Since Q and D are nonnegative,

$$1 + \frac{Q}{H^2U^2(-)D(-)} \geq 1 \tag{4-22}$$

Thus, Eq. (4-21) can be written:

$$\left[D(-) - \left(\frac{1}{a}\right)\underset{\sim}{v}\underset{\sim}{v}^T\right] \geq D(-)\left[1 - \frac{1}{1}\right] = 0 \tag{4-23}$$

Or, simply:

$$\left[D(-) - \left(\frac{1}{a}\right)\underset{\sim}{v}\underset{\sim}{v}^T\right] \geq 0 \tag{4-24}$$

Clearly, from Eq. (4-24), the assumption of positive, semidefiniteness (and, trivially, symmetry) is founded for at least the scalar case. The proof can be extended to the general case in a straightforward manner and will not be included here (8:656-658; 32:216).

Accepting the factorization as given in Eq. (4-19), the covariance update equation [Eq. (4-18)] becomes:

$$U(+)D(+)U^T(+) = U(-)\left[\bar{U}\bar{D}\bar{U}^T\right]U^T(-) \tag{4-25}$$

Or, regrouping the right-hand-side:

$$U(+)D(+)U^T(+) = \left[U(-)\bar{U}\right]\bar{D}\left[U(-)\bar{U}\right]^T \tag{4-26}$$

$U(-)$ and $\bar{U}$ are both unitary upper triangular, so the following equivalences can be made (31:393):

$$U(+) = U(-)\bar{U} \tag{4-27a}$$

$$D(+) = \bar{D} \tag{4-27b}$$

Thus, the covariance update equation has essentially been reduced to the factoring of a positive, semidefinite, symmetric matrix as given in Eq. (4-19).

Assuming $U(-)$ and $D(-)$ are available, Maybeck cites an efficient computational algorithm (originally due to Bierman) for computing $U(+)$ and $D(+)$ (32:394). This algorithm is given below as Eq. (4-28).

$$\hat{\underset{\sim}{f}} = U^T(-)H^T$$

$$v_j = D_{jj}(-)\hat{\underset{\sim}{f}}_j \qquad j = 1,2,\ldots,n$$

$$a_0 = Q$$

$$a_1 = a_0 + \hat{f}_1 v_1$$

$$D_{11}(+) = D_{11}(-)\left(\frac{a_0}{a_1}\right)$$

$$b_1 = v_1$$

$$\left.\begin{aligned} a_k &= a_{k-1} + \hat{f}_k v_k \\ D_{kk}(+) &= D_{kk}(-)\left(\frac{a_{k-1}}{a_k}\right) \\ b_k &= v_k \\ p_k &= -\left(\frac{f_k}{a_{k-1}}\right) \\ \left.\begin{aligned} U_{jk}(+) &= U_{jk}(-) + b_j p_k \\ b_j &= b_j + U_{jk}(-)v_k \end{aligned}\right\} & \quad j = 1,2,\ldots(k-1) \end{aligned}\right\} \quad k = 2,3,\ldots,n \tag{4-28}$$

The algorithm given by Eq. (4-28) constitutes the covariance update equation while the state update equation is still given by Eqs. (4-6) and (4-7). Note, in ill-conditioned problems, the Kalman gain used in Eq. (4-6b) is best obtained from

$$K = \frac{\underset{\sim}{b}}{a_n} \tag{4-29}$$

once the algorithm above has been completed (32:394). The state and covariance propagation equations from t_i^+ to t_{i+1}^- are yet to be cited.

The state propagation equation [Eq. (4-9)] remains unchanged:

$$\hat{\underset{\sim}{x}}(t_{i+1}^-) = \underset{\sim}{f}\left[\hat{\underset{\sim}{x}}(t_i^+), t_{i+1}\right] \tag{4-30}$$

As a first step in deriving a factored form of the covariance propagation equation, Eq. (4-10) is factored to give:

$$P(t_{i+1}^-) = \Phi(t_{i+1}, t_i)U(t_i^+)D(t_i^+)U^T(t_i^+)\Phi^T(t_{i+1}, t_i) + CQ_n(\Delta t)C^T \tag{4-31}$$

Without loss of generality, Q_n can be assumed to be a diagonal matrix. This is because the matrices C and Q_n can be regarded as the U-D factors of a nondiagonal (but symmetric) noise matrix (32:396).

It is easily verified by direct substitution that Eq. (4-30) can be written as

$$P(t_{i+1}^-) = Y(t_{i+1}^-)\tilde{D}(t_{i+1}^-)Y^T(t_{i+1}^-) \tag{4-32a}$$

where:

$$Y(t_{i+1}^-) = \left[\Phi(t_{i+1}, t_i)U(t_i^+) \;\vdots\; C\right] \tag{4-32b}$$

$$\tilde{D}(t_{i+1}^-) = \left[\begin{array}{c|c} D(t_i^+) & 0 \\ \hline 0 & Q_n(\Delta t) \end{array}\right] \tag{4-32c}$$

Note that Y is $n \times 2n$ and $\bar{D}$ is $2n \times 2n$ if C is assumed to be $n \times n$.

To keep the covariance in factored form, it is necessary to apply Eq (4-31a) such that:

$$P(t_{i+1}^-) = U(t_{i+1}^-)D(t_{i+1}^-)U^T(t_{i+1}^-) \tag{4-33}$$

Maybeck cites an efficient computational algorithm (originally due to Bierman) for obtaining the factors $U(t_{i+1}^-)$ and $D(t_{i+1}^-)$ from Eqs. (4-32) (32:396-397). The procedure can be given as:

$$\text{Let } \left[\underset{\sim}{a}_1 \vdots \underset{\sim}{a}_2 \vdots \cdots \vdots \underset{\sim}{a}_n\right] = Y^T(t_{i+1}^-)$$

$$\left.\begin{aligned} &c_{jk} = \bar{D}_{jj}(t_{i+1}^-)a_{jk} \quad j = 1,2,\ldots,n \\ &D_{kk}(t_{i+1}^-) = \underset{\sim}{a}_k^T \underset{\sim}{c}_k \\ &\underset{\sim}{d}_k = \frac{\underset{\sim}{c}_k}{D_{kk}(t_{i+1}^-)} \\ &\left.\begin{aligned} &U_{jk}(t_{i+1}^-) = \underset{\sim}{a}_j^T \underset{\sim}{d}_k \\ &\underset{\sim}{a}_j = \underset{\sim}{a}_j - U_{jk}(t_{i+1}^-)\underset{\sim}{a}_k \end{aligned}\right\} j = 1,2,(k-1) \end{aligned}\right\} k = n,(n-1),\ldots 2$$

$$\begin{aligned} &c_{j1} = \bar{D}_{jj}(t_{i+1}^-)a_{j1} \quad j = 1,2,\ldots 2n \\ &D_{11}(t_{i+1}^-) = \underset{\sim}{a}_1^T \underset{\sim}{c}_1 \end{aligned} \tag{4-34}$$

In this notation, c_{jk} is element j of column vector k. A similar statement can be made about the n column vectors $\underset{\sim}{a}_k$. Upon completing the procedure above, the propagated covariance can be assembled and inspected by multiplying out Eq. (4-33) if desired. But, the filter should not employ this assembled form when precision is important.

Finally, with all of the necessary equations at hand, a computer-style algorithm for the iterated, extended form of the U-D filter can now be presented. This is given in Figure 4-2. Notice, the filter iterates on $U(+)$ and $D(+)$ simultaneously with the state. This is required to obtain the Kalman gain in the most numerically precise way [Eqs. (4-28) and (4-29)]. *If* the covariance update could be moved out of the iteration loop somehow, then the total number of computations in the algorithm could be reduced.

In the problem studied, there was enough numerical stability in the update step to allow the Kalman gain to be calculated by

$$K = \frac{P(-)H^T}{a} \tag{4-35}$$

instead of Eq. (4-29). In fact, the difference in gain found by the two methods was identical to 13 - 14 digits once convergence was achieved. Thus, it was experimentally determined the filter could be slightly modified for enhanced speed as shown in Figure 4-3. It must be emphasized that the above modification to the U-D filter will, in general, reduce the numerical precision but does not *in this particular situation*. When in doubt, Figure 4-2 presents the preferred implementation.

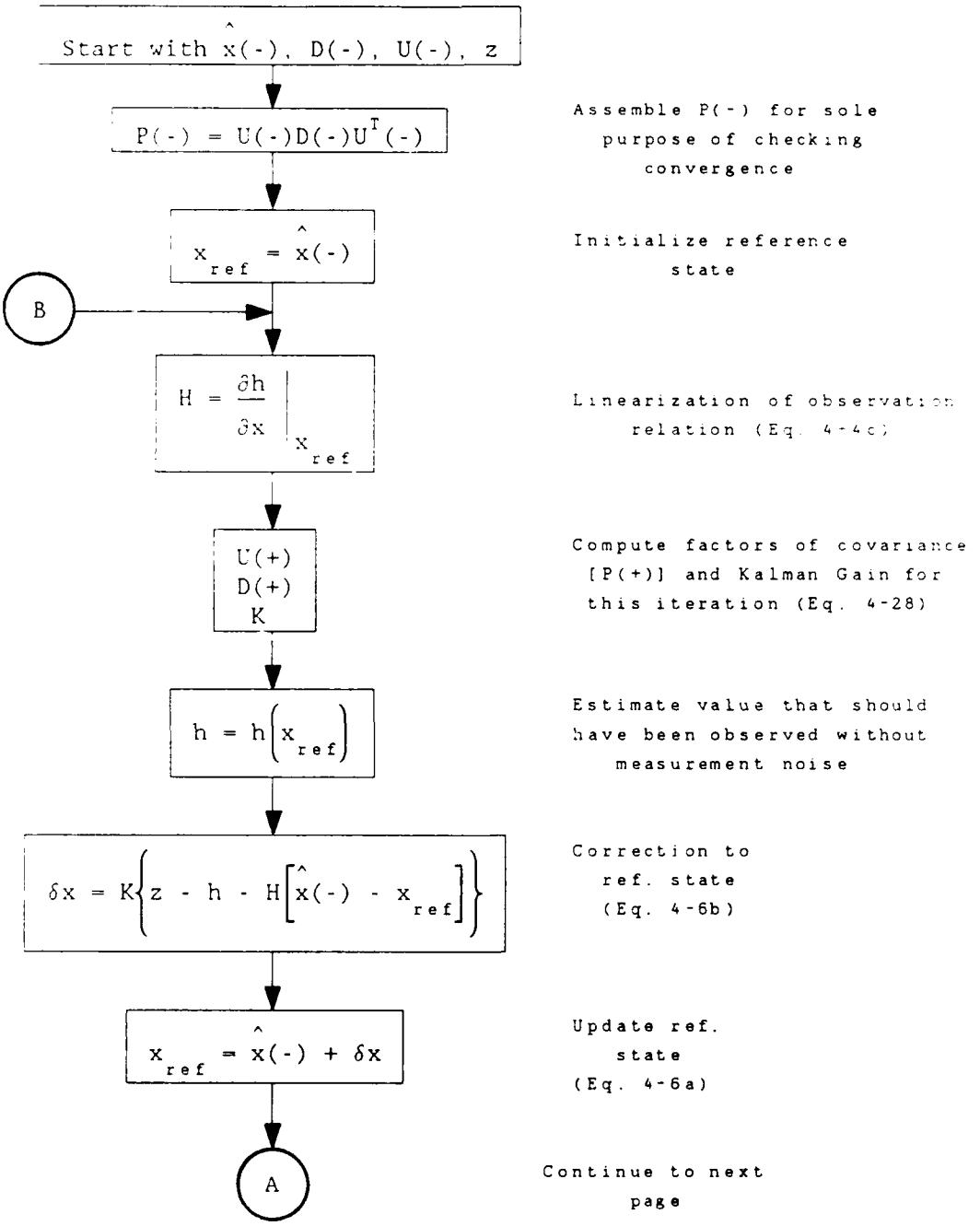

Figure 4-2. Algorithm for Iterated, Extended U-D Filter (Part 1 of 2)

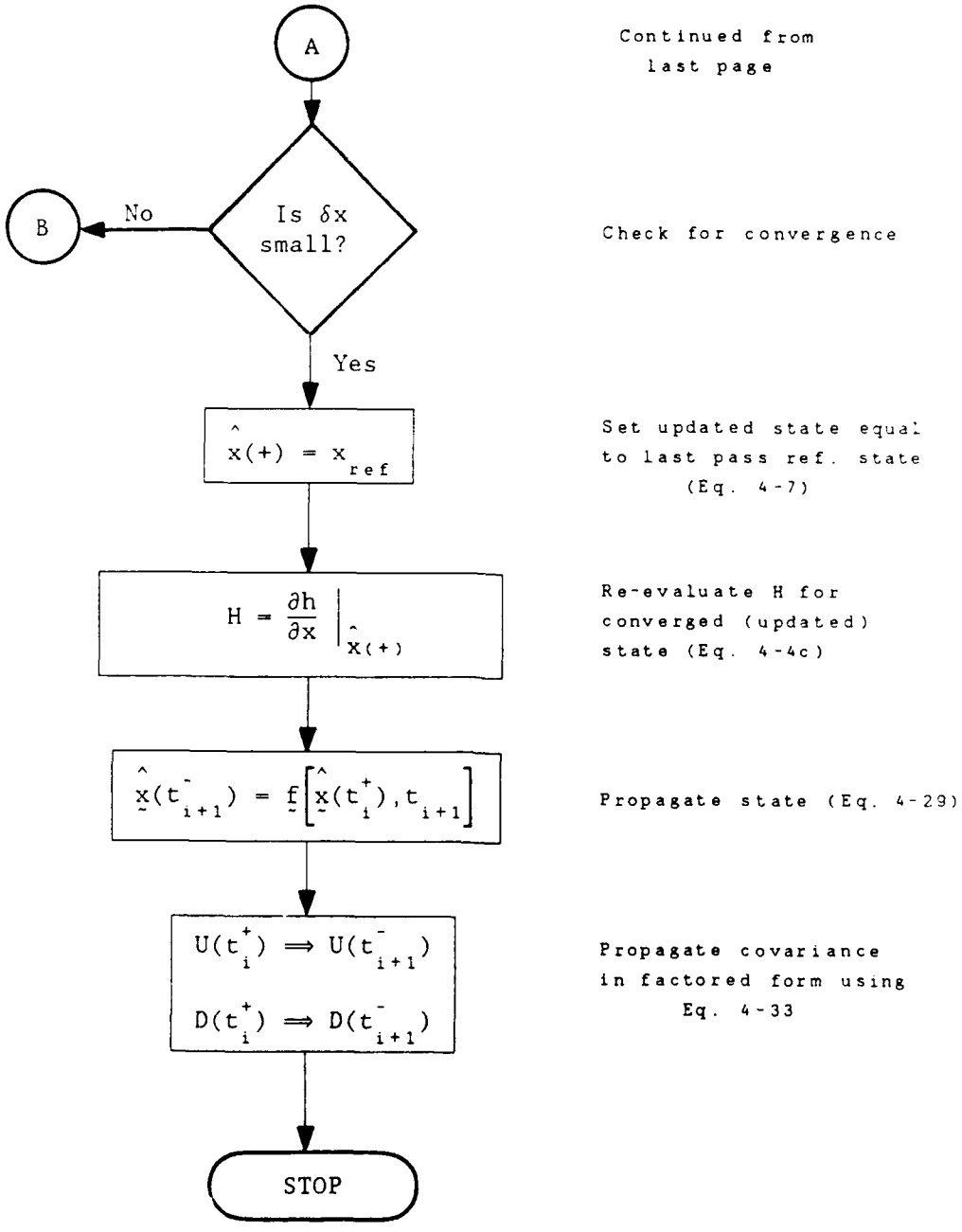

Figure 4-2. Algorithm for Iterated, Extended U-D Filter (Part 2 of 2)

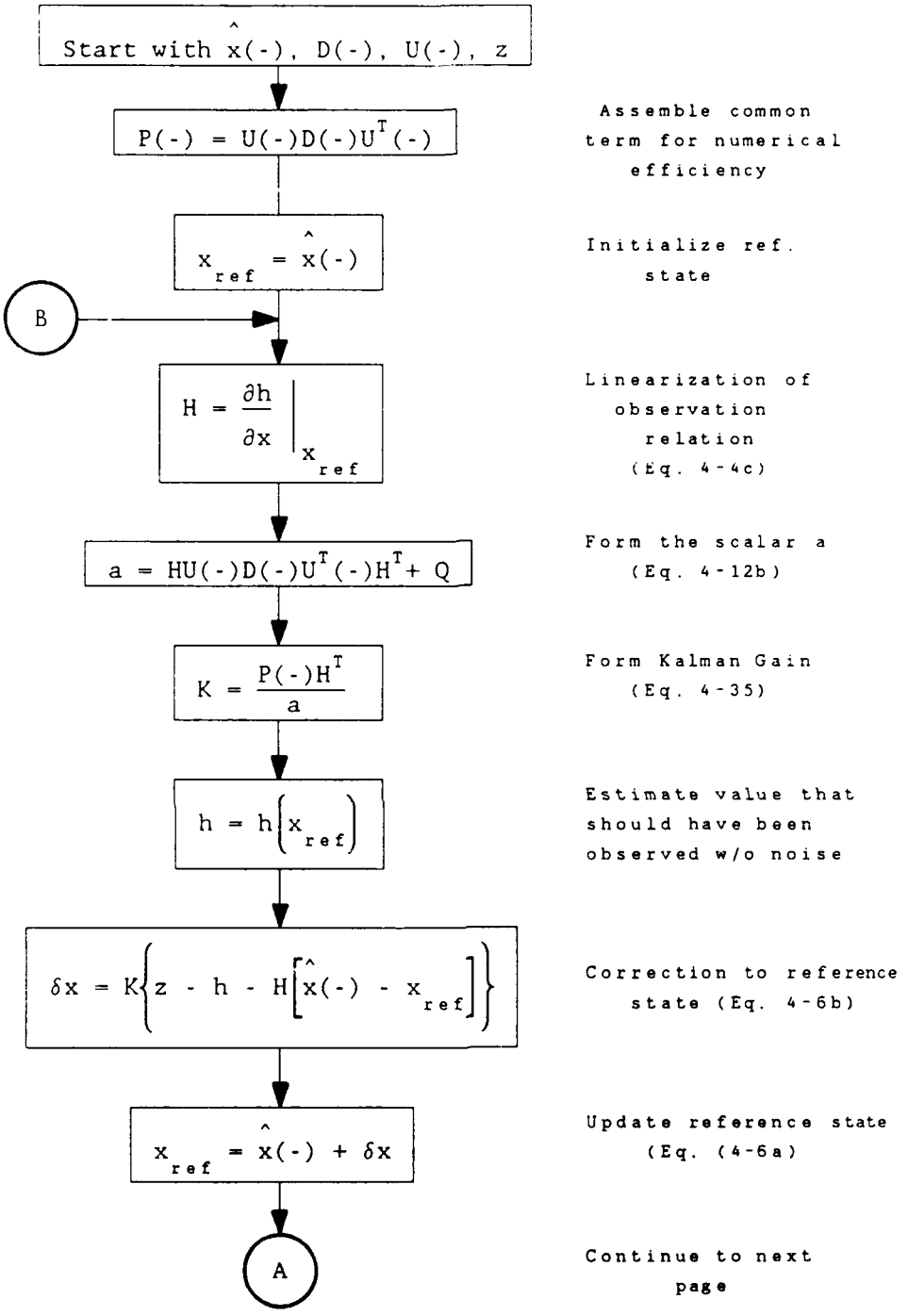

Figure 4-3. Algorithm for Iterated, Extended U-D Filter (Modified for Execution Speed) (Part 1 of 2)

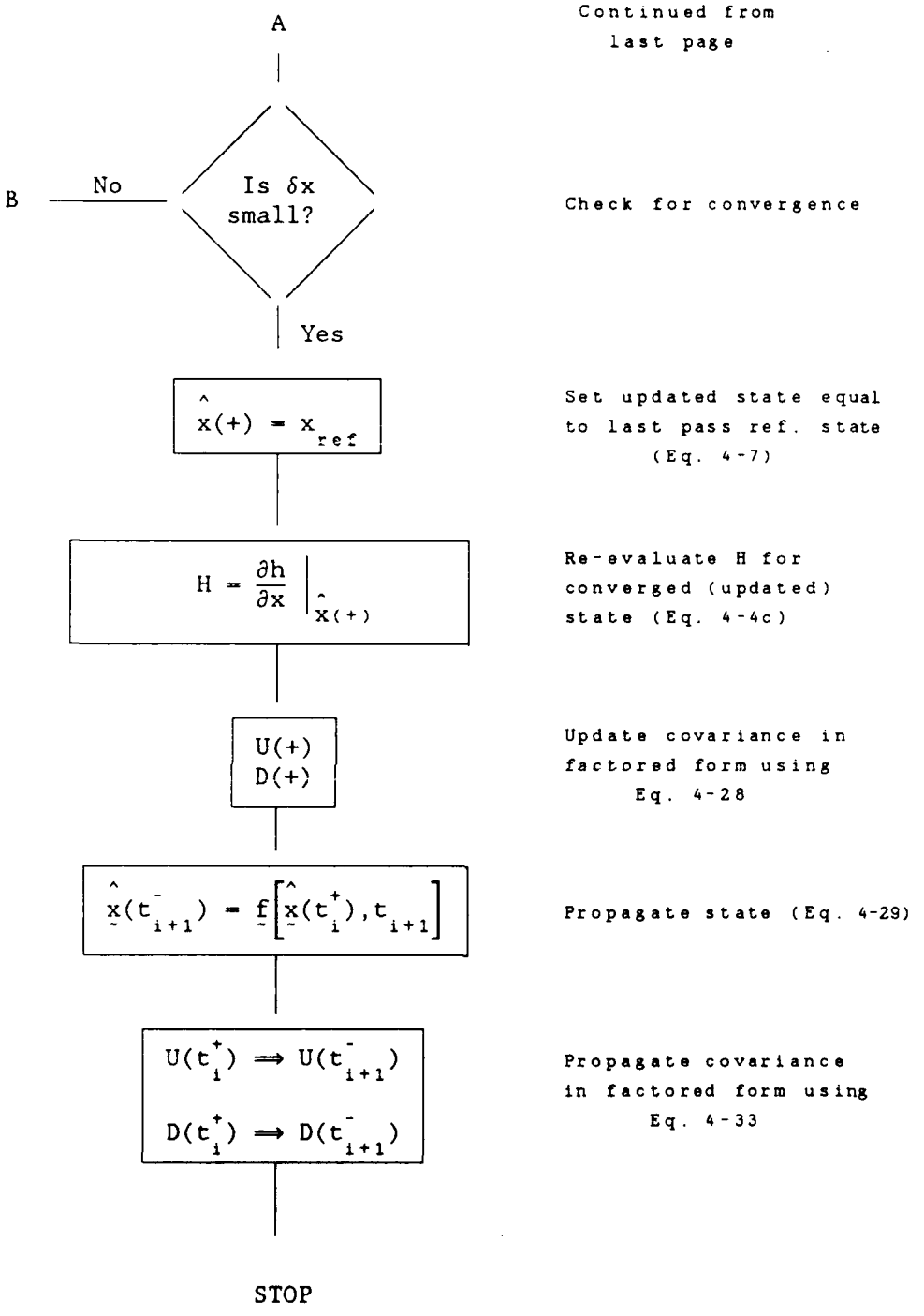

Figure 4-3. Algorithm for Iterated, Extended U-D Filter (Modified for Execution Speed) (Part 2 of 2)

When dealing with imperfect instruments in real-world situations, the possibility that some measurements will be greatly in error exists. It is, therefore, desirable to provide some sort of mechanism to reject "bad data" and minimize their effect on the estimated parameters. Two schemes are immediately applicable: residual checking and maximum iteration limits.

The first, residual checking, compares the actual observations, $z(t_i)$, to those expected, $h\left[\hat{\underset{\sim}{x}}(t_i^-)\right]$. If these differ more than a predetermined amount, then that datum is ignored and the state, $\hat{\underset{\sim}{x}}(t_i^-)$, propagated to the next observation time. This method requires an extra, albeit trivial, computation of $\left|z(t_i) - h\left[\hat{\underset{\sim}{x}}(t_i^-)\right]\right|^2$ at each observation. A maximum iteration scheme, on the other hand, sets a limit on the number of times the filter is allowed to iterate before giving up. If this limit is exceeded, that datum is ignored and the state propagated to the next datum.

The latter scheme, maximum iteration limits, provides protection against bad data in addition to solving another problem inherent to many iteration schemes. Due to numerical truncation, state estimates may oscillate about a solution. Including a limit on the number of iterations prevents an "infinite loop" from developing; in fact, a maximum iteration check must be incorporated even when applying residual checking. In this research, it was only necessary to limit the iterations while ignoring the residual. For reference, the maximum number of iterations was set to seven for all orbits. This number was selected by trial and error to be more than twice as large as the typical number of iterations required for convergence.

For the navigation problem at hand, this filter can be directly applied once a dynamics model $\underset{\sim}{f}$ has been selected and the observation quantities in Eq. (4-2) derived.

The Dynamics Model

As mentioned, it is necessary to have a dynamics model available to propagate the state estimate forward from one time to another. Algebraically, this is given by:

$$\underset{\sim}{x}(t_i) = \underset{\sim}{f}\left[\underset{\sim}{x}(t_o), t_i\right] \tag{4-36}$$

Note that process noise is not shown in Eq. (4-36). The previous section only added process noise for the linearization of the dynamics. This, in essence, says the dynamics are completely deterministic and the majority of the uncertainty is introduced in the linearization. Further, any errors that actually do exist in $\underset{\sim}{f}$ are accounted for when the filter is tuned. Two approaches exist for obtaining dynamical equations of the form above: numerical integration of differential equations of motion and closed form (general perturbations) solutions. Regardless of which method is used, real-time applications require this propagation to be a "fast" operation.

Many previous studies have used numerical integration (11; 17; 22; 34; 39; 42). Precise numerical integration of dynamical equations requires more and more computations as the time between observations grows. For real-time operations on small computers, this can be

unacceptable because the filter might not have time to complete one update before the next datum becomes available (44:77,118).

General perturbations methods, on the other hand, require the same number of computations regardless of the time step involved. This advantage does not come free; perturbation theories are closed-form solutions based on small deviations from a reference orbit and remain valid only for finite time spans. Thus, the reference orbit must be reinitialized periodically, with the time between required reinitializations depending on the model. This does not pose a problem in the current application; however. Each time a new datum is incorporated and the state estimate updated, the reference orbit is reinitialized automatically. Therefore, a general perturbations method is completely appropriate here. The specific theory used in this research is based on that derived by Aksnes (2; 3; 4).

Aksnes' model is a reformulation of Brouwer's first-order satellite theory in terms of the Hill variables (4:70; 12). This change of variables allows the method to be applied even where Brouwer's theory breaks down -- zero eccentricity orbits. Like Brouwer's, this theory includes the J_2 - J_5 zonal harmonics of the primary and ignores air drag (4:70; 12). Between observations, the effects of drag cannot be separated from the imperfections in the observations and the model is still justified as derived by Aksnes. Aksnes has shown his algorithm predicts satellite positions to within 60 m of positions predicted via numerical integration of the corresponding differential equations (2:1075-1076; 3:32;

4:70,74). This precision is obtained for a period of at least 6 days and greatly reduces the number of computations involved (4:70).

The algorithm as derived by Aksnes accepts input and produces output in two sets of state variables: a set of classical orbital elements and a set of cartesian position and velocity elements in the geocentric-equatorial frame. As will be seen when the observation relations are introduced, values in the latter system must be available. The former set, though, was replaced with the better defined equinoctial elements (8:490-495; 27:44; 28:3-8 - 3-9; Appendix C). The reason for this change is quite subtle and strictly for numerical reasons.

The filter employed relies on iterations about a state near the correct one; therefore, the state variables should "behave" in a "rational" way. In other words, a small change in one variable should not produce an overly large change in another during the course of iterating about a solution. This is not always the case for the classical orbital elements; for example, near an inclination of zero ($i \approx 0$), the longitude of the ascending node, Ω, becomes ill-defined. Small errors in the other elements can cause Ω to vary by as much as π between iterations. Similar statements can be made about the argument of perigee, ω, when the eccentricity, e, approaches zero. Since the iterated, extended U-D filter formulation relies on "rational" behavior, this is completely unacceptable.

Many of the useful orbits fall into situations where one or more of the classical elements are ill-defined, so this research chose to convert Aksnes' classical elements into a set of equinoctial elements.

These elements are well-defined for all orbits studied. In terms of the classical orbital elements they are:

$$a_f = e \cos(\Omega + \omega) \tag{4-37a}$$

$$a_g = e \sin(\Omega + \omega) \tag{4-37b}$$

$$a = a \tag{4-37c}$$

$$L = \Omega + \omega + M \tag{4-37d}$$

$$\chi = \frac{\sin i \sin\Omega}{1 + \cos i} \tag{4-37e}$$

$$\psi = \frac{\sin i \cos\Omega}{1 + \cos i} \tag{4-37f}$$

where a is the semimajor axis; M is the mean anomaly; L is the mean longitude; e, i, Ω, and ω have already been defined; and a_f, a_g, χ, and ψ do not have formal names (8:490-492; 26:44). A manner in which this the change of variables can be incorporated is shown in Figure 4-4.

The filter requires the computation of the state transition matrix

$$\Phi(t_{i+1}, t_i) = \left.\frac{\partial f}{\partial x_i}\right|_{x_{ref}} \tag{4-38}$$

at each observation point to propagate the covariance matrix forward [Eq. (4-31)]. The complexity of Aksnes' equations, $\underset{\sim}{f}$, essentially rules out an analytical computation of this matrix in real time. Numerical precision in this step is not essential, however. Some formulations even treat the matrix as a time-invariant constant (32:324). In this

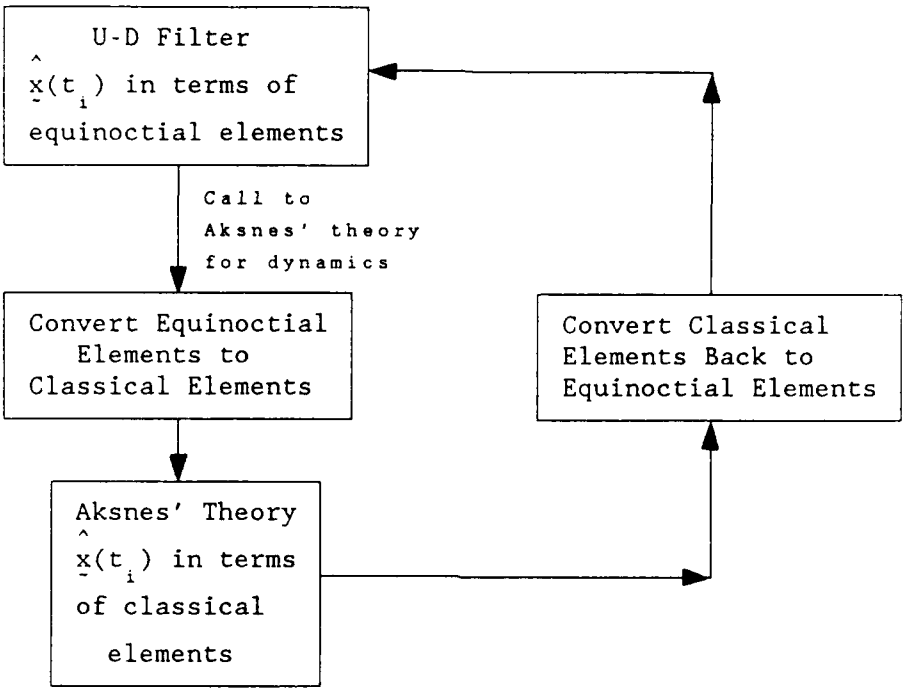

Figure 4-4. Orbital Element Conversion Routine

study, it was found that Φ could be numerically approximated through one-sided numerical differentiation. Approximations for Φ obtained in this manner still produced propagated covariance matrices, P(-), which were representative of the true errors in the states. For completeness, note the form of the state transition matrix is:

$$\Phi(t_{i+1},t_i) = \begin{bmatrix} \frac{\partial a_{f_{i+1}}}{\partial a_{f_i}} & \frac{\partial a_{f_{i+1}}}{\partial a_{g_i}} & \frac{\partial a_{f_{i+1}}}{\partial a_i} & \frac{\partial a_{f_{i+1}}}{\partial L_i} & \frac{\partial a_{f_{i+1}}}{\partial \chi_i} & \frac{\partial a_{f_{i+1}}}{\partial \psi_i} \\ \frac{\partial a_{g_{i+1}}}{\partial a_{f_i}} & \frac{\partial a_{g_{i+1}}}{\partial a_{g_i}} & \frac{\partial a_{g_{i+1}}}{\partial a_i} & \frac{\partial a_{g_{i+1}}}{\partial L_i} & \frac{\partial a_{g_{i+1}}}{\partial \chi_i} & \frac{\partial a_{g_{i+1}}}{\partial \psi_i} \\ \frac{\partial a_{i+1}}{\partial a_{f_i}} & \frac{\partial a_{i+1}}{\partial a_{g_i}} & \frac{\partial a_{i+1}}{\partial a_i} & \frac{\partial a_{i+1}}{\partial L_i} & \frac{\partial a_{i+1}}{\partial \chi_i} & \frac{\partial a_{i+1}}{\partial \psi_i} \\ \frac{\partial L_{i+1}}{\partial a_{f_i}} & \frac{\partial L_{i+1}}{\partial a_{g_i}} & \frac{\partial L_{i+1}}{\partial a_i} & \frac{\partial L_{i+1}}{\partial L_i} & \frac{\partial L_{i+1}}{\partial \chi_i} & \frac{\partial L_{i+1}}{\partial \psi_i} \\ \frac{\partial \chi_{i+1}}{\partial a_{f_i}} & \frac{\partial \chi_{i+1}}{\partial a_{g_i}} & \frac{\partial \chi_{i+1}}{\partial a_i} & \frac{\partial \chi_{i+1}}{\partial L_i} & \frac{\partial \chi_{i+1}}{\partial \chi_i} & \frac{\partial \chi_{i+1}}{\partial \psi_i} \\ \frac{\partial \psi_{i+1}}{\partial a_{f_i}} & \frac{\partial \psi_{i+1}}{\partial a_{g_i}} & \frac{\partial \psi_{i+1}}{\partial a_i} & \frac{\partial \psi_{i+1}}{\partial L_i} & \frac{\partial \psi_{i+1}}{\partial \chi_i} & \frac{\partial \psi_{i+1}}{\partial \psi_i} \end{bmatrix} \tag{4-39}$$

In summary, the dynamics model employed by the U-D filter is a modified version of Aksnes' first order theory. His theory has been modified such that input and output variables are a set of equinoctial elements as well as the geocentric-equatorial position and velocity. The matrix Φ required to propagate the covariance forward from one time to another is computed numerically and has the form given in Eq. (4-39).

The Observations

The essence of any navigation system is the ability to use observed measurements to estimate one's position. For a star sensor/Earth sensor

combination, relationships between observations and estimated position are easily derived. It is also highly desirable to know the precision of the measurements; indeed, this is a requirement to apply the U-D filter. This can be derived as a function of given instrument accuracies.

Two Earth horizon sensors with their field-of-view offset but in the same plane can detect and track two opposite edges of the Earth to create horizon vectors. By bisecting these vectors, a line between the center of the Earth and the satellite can be drawn. This is called the observed local vertical vector and may differ from the true local vertical by an angle $\delta\theta$, as shown in Figure 4-5. $\delta\theta$ is a measure of the precision of the horizon sensors.

A star sensor can establish a line-of-sight vector to a known star. The direction of this vector relative to the observed local vertical can be easily measured. As with horizon sensors, this measurement is not perfect. Figure 4-5 shows the assumed error $\delta\psi$ in this measurement.

For mathematical convenience, the observation to be used is not a set of angular measurements but rather the cosine of the included angle between the observed local vertical and the line-of-sight to the star. Thus, if $\hat{\gamma}$ is the observed included angle, $z = \cos\hat{\gamma}$ is the data recorded by the navigation system.

Since the observation is not perfect, its relationship to the true included angle $\gamma = \hat{\gamma} - \delta\theta + \delta\psi$ can be written as:

$$\cos\hat{\gamma} = \cos\gamma + v(t_i) \qquad (4\text{-}40)$$

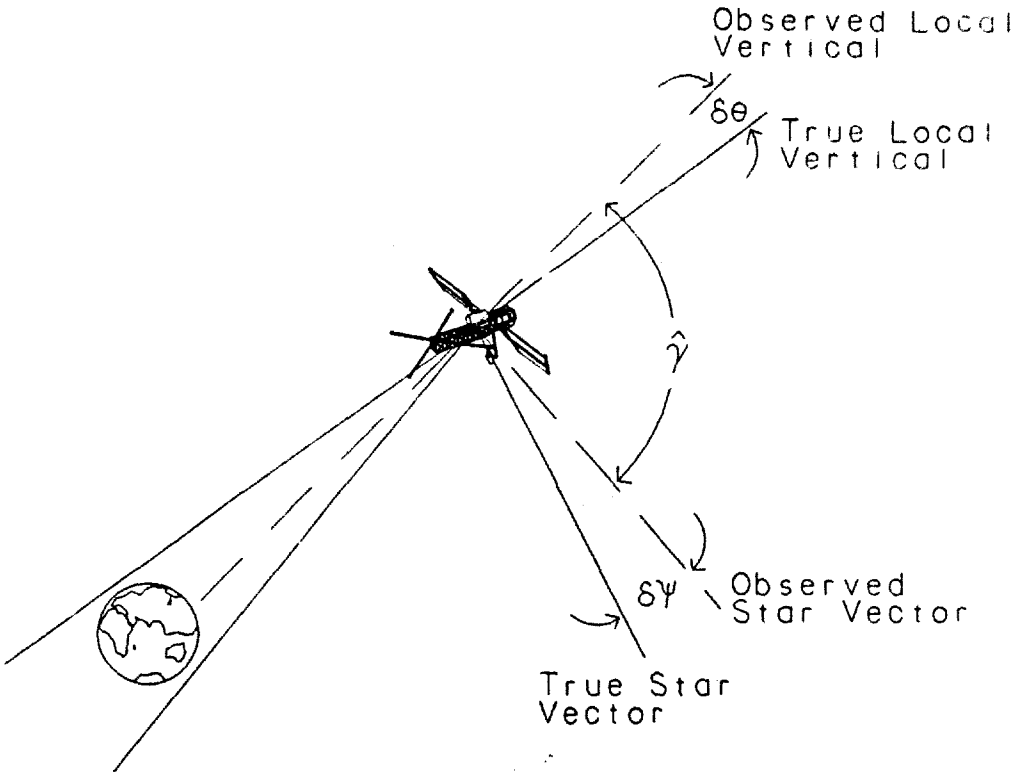

Figure 4-5. Observation Geometry

$v(t_i)$ can be considered to be zero-mean discrete-time white Gaussian noise because real instruments typically exhibit such statistics (32:9; 44:5; 45). It is the function of an estimator to minimize the effect of the noise in these measurements (32:4-5).

To be consistent with notation in the filter derivation, this is rewritten as:

$$z(t_i) = h\left[\underline{x}(t_i)\right] + v(t_i) \qquad (4\text{-}41)$$

h is termed the observation relation and is a function of the state variables $\underset{\sim}{x}$.

To relate the position to a perfect observation, $z(t_i) = h\left[\underset{\sim}{x}(t_i)\right] = \cos\left[\gamma(t_i)\right]$, consider measuring positions with respect to the geocentric-equatorial system. The satellite's position vector can then be written

$$\underset{\sim}{r} = x\hat{e}_x + y\hat{e}_y + z\hat{e}_z \tag{4-42}$$

where $\hat{e}_x$, $\hat{e}_y$, $\hat{e}_z$ are unit vectors. $\hat{e}_x$ points in the direction of the vernal equinox; $\hat{e}_z$ points out the Earth's north pole; $\hat{e}_y$ completes the system such that it lies in the Earth's equatorial plane and $\hat{e}_z = \hat{e}_x \times \hat{e}_y$. In this coordinate system, the position vector of the satellite lies along the local vertical. Therefore, while the components of $\underset{\sim}{r}$ (x, y, and z) are unknown, their magnitudes relative to each other are approximately known because the position vector is a scalar multiple of the local vertical vector. This implies that information about the local vertical can be used in determining the position vector. The manner in which this information enters into position determination is derived presently.

A vector pointing from Earth in the direction of any star is essentially fixed in this coordinate system; that is, it remains unchanged even when viewed from opposite sides of the Earth's orbit around the sun (7:56-57). Thus, a star's absolute position relative to the geocentric-equatorial system can be considered to be a known constant vector,

regardless of the satellite's position. Then, for any given star, there is a known unit vector that gives its direction:

$$\hat{e}_s = x_s \hat{e}_x + y_s \hat{e}_y + z_s \hat{e}_z \tag{4-43}$$

Employing the definition of the dot product between the local vertical and the line-of-sight to the star yields

$$h = \cos\gamma = \frac{\underset{\sim}{r} \cdot \hat{e}_s}{|\underset{\sim}{r}||\hat{e}_s|} \tag{4-44}$$

for any given time. Substituting Eqs. (4-42) and (4-43) into Eq. (4-44):

$$h = \cos\gamma = \frac{xx_s + yy_s + zz_s}{\sqrt{x^2 + y^2 + z^2}} \tag{4-45}$$

The known values in this equation are x_s, y_s, and z_s, while the unknowns are x, y, and z. Further, an approximation to h is the known observation z. Thus, each star sighting can be used to obtain one component of the position (with the filter minimizing the effect of the error between h and z).

It remains to be shown how to relate errors in the sensor measurements to errors in the observed quantity $z = \cos\hat{\gamma}$. Indeed, this is a relationship required by the filter. To derive this, refer back to Figure 4-5. The observed included angle is $\hat{\gamma}$ while the true included

angle is $\gamma = \hat{\gamma} - \delta\theta + \delta\psi$; therefore, the error in the measurement is found by subtracting:

$$|\delta z| = |\cos\gamma - \cos\hat{\gamma}| = |\cos(\hat{\gamma} - \delta\theta + \delta\psi) - \cos\hat{\gamma}| \tag{4-46}$$

Expanding out the first term on the right-hand-side:

$$|\delta z| = |\cos(-\delta\theta + \delta\psi)\cos\hat{\gamma} - \sin(-\delta\theta + \delta\psi)\sin\hat{\gamma} - \cos\hat{\gamma}| \tag{4-47}$$

Since $\delta\theta$ and $\delta\psi$ are, presumably, small pointing errors, the small angle assumptions

$$\cos(-\delta\theta + \delta\psi) \approx 1 \tag{4-48a}$$

$$\sin(-\delta\theta + \delta\psi) \approx -\delta\theta + \delta\psi \tag{4-48b}$$

can be introduced. This reduces Eq. (4-46) to:

$$|\delta z| = |-(-\delta\theta + \delta\psi)\sin\hat{\gamma}| \tag{4-49}$$

Squaring both sides and recalling that the observed quantity is $z = \cos\hat{\gamma}$:

$$(\delta z)^2 = \left[(\delta\theta)^2 - 2(\delta\theta\delta\psi) + (\delta\psi)^2\right]\left(1 - z^2\right) \tag{4-50}$$

Finally, the standard conditional expectation operator (32:95; 44:10-12) can be applied to both sides to yield

$$\sigma_z^2 = \left(\sigma_\theta^2 + \sigma_\psi^2\right)\left(1 - z^2\right) = Q \tag{4-51}$$

where, since θ and ψ are measured by different instruments, it has been assumed that $\delta\theta$ and $\delta\psi$ are statistically independent. Note, the term $\left(1 - z^2\right)$ comes through the operation unchanged because the expected value of a measurement is the true value (44:12). σ_z, σ_θ, and σ_ψ are the standard deviations of the measurement, the horizon sensors, and the star sensor, respectively. σ_θ and σ_ψ are given by sensor manufacturer or can be determined by experiment. The term $\sigma_\theta^2 + \sigma_\psi^2$ can be considered to be a measure of the composite instrument variance, $\sigma_{comp}^2 = \sigma_\theta^2 + \sigma_\psi^2$.

σ_z is the term that will be needed for the filter to determine the accuracy of the position estimate. The information contained in σ_z is used by the navigation filter to produce standard deviation values for the state estimates. Once Earth and star sensors are selected, this number depends only on the observation z since $z = z\left(\sigma_{comp}, z^2\right)$ as given in Eq. (4-51).

Since at any point the modified Aksnes' theory has available the geocentric-equatorial coordinates as well as the equinoctial elements, the observation relation [Eq. (4-45)] and its variance [Eq. (4-51)] can be readily evaluated. The estimator, through Eq. (4-4c), requires the computation of the row vector

$$H = \frac{\partial h}{\partial \underline{x}} \tag{4-52}$$

where $\underline{x}$ is the state *in use by the estimator*, in this case the equinoctial elements. Thus, the chain rule must be applied when forming

Eq. (4-52) from Eq. (4-45). Designating the coordinates in the geocentric-equatorial system as $\underset{\sim}{x}_{g-e}$ and the equinoctial elements as simply $\underset{\sim}{x}$, the row vector H can be written as:

$$H = \left[\frac{\partial h}{\partial \underset{\sim}{x}_{g-e}}\right]\left[\frac{\partial \underset{\sim}{x}_{g-e}}{\partial \underset{\sim}{x}}\right] \tag{5-53}$$

The term $\left[\frac{\partial h}{\partial \underset{\sim}{x}_{g-e}}\right] = \left[\frac{\partial h}{\partial x}\ \frac{\partial h}{\partial y}\ \frac{\partial h}{\partial z}\ \frac{\partial h}{\partial \dot{x}}\ \frac{\partial h}{\partial \dot{y}}\ \frac{\partial h}{\partial \dot{z}}\right]$ can be analytically calculated from Eq. (4-45)

$$\left[\frac{\partial h}{\partial \underset{\sim}{x}_{g-e}}\right] = \begin{bmatrix} \dfrac{x_s r^2 - x\left(xx_s + yy_s + zz_s\right)}{r^3} \\ \dfrac{y_s r^2 - y\left(xx_s + yy_s + zz_s\right)}{r^3} \\ \dfrac{z_s r^2 - z\left(xx_s + yy_s + zz_s\right)}{r^3} \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \tag{4-54}$$

with $x, y, z, \dot{x}, \dot{y}, \dot{z}$ being the geocentric-equatorial position and velocity components. $r = \sqrt{x^2 + y^2 + z^2}$ is the orbital radius. The matrix $\left[\frac{\partial \underset{\sim}{x}_{g-e}}{\partial \underset{\sim}{x}}\right]$ is best calculated though numerical differentiation of

the modified Aksnes theory. For completeness, though, note that this term is of the form

$$\left[\frac{\partial \underset{\sim}{x}_{g-e}}{\partial \underset{\sim}{x}}\right] = \begin{bmatrix} \frac{\partial x}{\partial a_f} & \frac{\partial x}{\partial a_f} & \frac{\partial x}{\partial a} & \frac{\partial x}{\partial L} & \frac{\partial x}{\partial \chi} & \frac{\partial x}{\partial \psi} \\ \frac{\partial y}{\partial a_f} & \frac{\partial y}{\partial a_g} & \frac{\partial y}{\partial a} & \frac{\partial y}{\partial L} & \frac{\partial y}{\partial \chi} & \frac{\partial y}{\partial \psi} \\ \frac{\partial z}{\partial a_f} & \frac{\partial z}{\partial a_g} & \frac{\partial z}{\partial a} & \frac{\partial z}{\partial L} & \frac{\partial z}{\partial \chi} & \frac{\partial z}{\partial \psi} \\ \frac{\partial x}{\partial a_f} & \frac{\partial x}{\partial a_g} & \frac{\partial x}{\partial a} & \frac{\partial x}{\partial L} & \frac{\partial x}{\partial \chi} & \frac{\partial x}{\partial \psi} \\ \frac{\partial y}{\partial a_f} & \frac{\partial y}{\partial a_g} & \frac{\partial y}{\partial a} & \frac{\partial y}{\partial L} & \frac{\partial y}{\partial \chi} & \frac{\partial y}{\partial \psi} \\ \frac{\partial z}{\partial a_f} & \frac{\partial z}{\partial a_g} & \frac{\partial z}{\partial a} & \frac{\partial z}{\partial L} & \frac{\partial z}{\partial \chi} & \frac{\partial z}{\partial \psi} \end{bmatrix} \tag{5-55}$$

where a_f, a_g, a, L, χ, and ψ are the equinoctial elements previously defined. Therefore, the row vector H can be easily assembled at any point once the terms in Eqs. (4-54) and (4-55) have been computed.

This subsection has derived all of the observation relations required by an autonomous navigation system. A more detailed analysis of this and similar methods for obtaining a celestial position fix is given by Battin (8:623-641).

Now that all of the necessary components of an autonomous navigation have been introduced (filter, dynamics model, and observation

relations), it is appropriate to present a summary of how these tie together.

Tying It All Together

The estimation problem at hand is addressed through the application of a factored form of the Kalman filter; in particular, an iterated, extended form of the Bierman-Thornton U-D factorization is used. The dynamics model selected for use by this filter is a modification of Aksnes' general perturbations model. Algebraically, this is given by

$$\underset{\sim}{x}(t_i) = \underset{\sim}{f}\left[\underset{\sim}{x}(t_o), t_i\right] \tag{4-56}$$

where the state $\underset{\sim}{x}$ is a set of equinoctial elements and t_o is an epoch time for which an estimate of the state is known.

The observation relations required by the filter are:

$$z_i = \cos\gamma_i + v_i \tag{4-57a}$$

$$h_i = \cos\gamma_i \tag{4-57b}$$

$$Q_i = \sigma^2_{z_i} = \sigma^2_{comp}\left(1 - z_i^2\right) \tag{4-57c}$$

γ_i is the included angle between the local vertical and the star sighted at time t_i. v_i is Gaussian noise with the properties previously described at t_i. The composite instrument variance, $\sigma^2_{comp} = \sigma^2_\theta + \sigma^2_\psi$, is a known value.

Together, the filter, dynamics model, and observations form a system capable of estimating an Earth-satellite's celestial position. The instrumentation required to obtain data is hardware currently on many existing satellites (42:14). Thus, the implementation of this scheme could be virtually immediate.

The accuracy and versatility of this configuration can be fully appreciated when demonstrated via accurate simulations. The next two chapters present Monte Carlo results for a standardized test orbit, low-Earth (US Space Station) orbit (15), semisynchronous (GPS/NAVSTAR) orbit (14:34-35; 17:69; 22:589; 30:3), and synchronous orbit.

V. System Characteristics

Before applying the system derived in the last chapter to specific spacecraft, it was useful to study as many aspects of the configuration as possible. This was accomplished by creating a "generic" test orbit and varying parameters of the filter and orbit individually. The baseline test orbit had the initial osculating classical orbital elements (7:58-60; 27:65; 37:25-28, 82; Appendix C):

$$a = 9000 \text{ km} \tag{5-1a}$$

$$e = 0.2 \tag{5-1b}$$

$$M = 10^\circ \tag{5-1c}$$

$$i = 30^\circ \tag{5-1d}$$

$$\omega = 40^\circ \tag{5-1e}$$

$$\Omega = 50^\circ \tag{5-1f}$$

Or, in terms of equinoctial elements (8:490-495; 27:44; 28:3-8 - 3-9; Appendix C):

$$a_f = 0 \tag{5-2a}$$

$$a_g = 0.2 \tag{5-2b}$$

$$a = 9000 \text{ km} \tag{5-2c}$$

$$L = 100^\circ \tag{5-2d}$$

$$\chi \approx 0.2053 \tag{5-2e}$$

$$\psi \approx 0.1722 \tag{5-2f}$$

This orbit was selected for two reasons. First, it is high enough to be virtually free from the effects of atmospheric drag, yet low enough to avoid significant third-body perturbations from the Sun and Moon. Second, this orbit is non-zero in all of the classical elements; therefore, the effects of driving certain terms to zero could be investigated.

For this baseline design, observations were taken approximately 13 1/2 minutes apart (a 0.00124 Hz data rate) with instruments accurate to 0.01° ($\sigma_{comp} = 0.01^\circ$). The star sighted was whichever star in the onboard catalog was nearest the orbit plane and not behind the Earth at any given time. (The 14 stars in the onboard catalog are listed in Table A-2 of Appendix A.) The standard deviations of the initial errors were 6 km and 6 km/week in each component of position and velocity, respectively -- well within the capabilities of traditional means at NCMC (42:63; 45). The baseline parameters of the satellite, filter , and orbit are summarized in Table 5-1.

It should be noted, however, that the initial covariance matrix given to the filter, $P(t=0^-)$, represented errors twice those cited above. This eliminated *all* numerical failures (division by zero, etc.) during the initial acquisition transient (filter initialization) by artificially deweighting a priori information. In fact, the trick of "lying" about the initial error by a factor of two was so successful that it was applied to all orbits studied, thereby making the system much more robust.

Table 5-1. Baseline Configuration		
Initial Orbital Elements:	Classical	Equinoctial
	$a = 9000$ km	$a_f = 0$
	$e = 0.2$	$a_g = 0.2$
	$M = 10^\circ$	$a = 9000$ km
	$i = 30^\circ$	$L = 100^\circ$
	$\omega = 40^\circ$	$\chi \approx 0.2053$
	$\Omega = 50^\circ$	$\psi \approx 0.1722$
True Initial Position Error:	10.4 km (1σ)	
True Initial Position Error:	10.4 km/week (1σ)	
Coeff. of Drag:	2.0	
Effective Mass:	2000 kg	
Effective Area:	10 m^2	
Data Rate:	0.00124 Hz	
σ_{comp}:	0.01°	
θ_{target}:	0° (In the orbit plane)	

Figure 5-1 shows the Monte Carlo results for a small portion of time over which the U-D filter was operated. Graphed is the actual (or "true") root mean square (RMS) value of the position error and the root mean square of the position error as computed by the filter [via the covariance matrix $P(t_i^+)$]. These curves represent the actual and filter-computed RMS errors [or standard deviations (σ)], respectively, as labelled in the figure (44:110-111). Periodic oscillations can be seen in both the actual and filter-computed values. This is a function

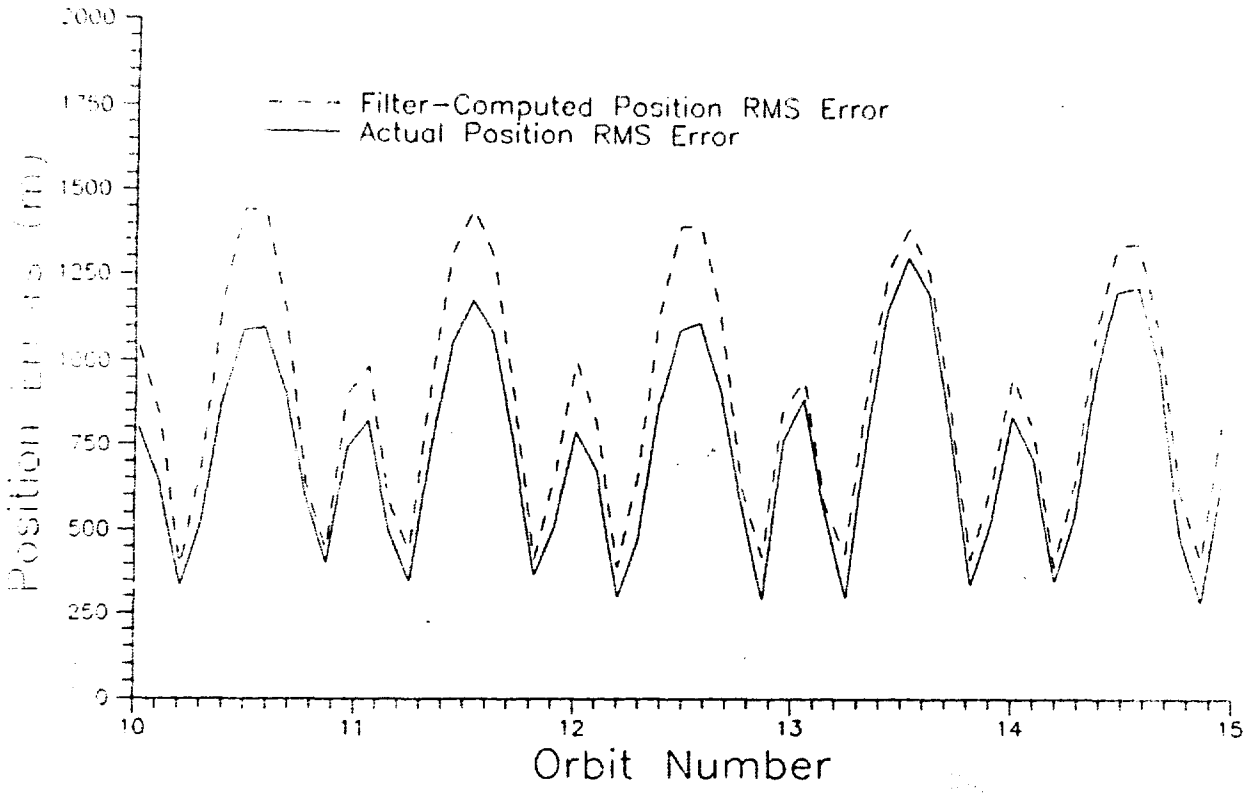

Figure 5-1. Monte Carlo Results for Test Orbit
(Short Timespan, Unsmoothed Output)

of Aksnes' model (4:74). When plotted over longer times, these variations tend to "blur" the graphs and distract from overall trends. The oscillations are regular with periods essentially equal to the orbital period; therefore, it is beneficial to average their values over an orbit to "smooth out" the appearance of the graph for presentation. To minimize the loss of information about this variation, it is also desirable to calculate the standard error of the mean over one period for these quantities

$$SEM = \frac{SSD}{\sqrt{N}} \qquad (5\text{-}2)$$

where SSD is the sample standard deviation (either the actual RMS error or the filter computed value, whichever applies) and N is the total number of points used in the computation of the SSD. For most of the remaining plots in this research, curves such as those in Figure 5-1 are replaced by two sets of lines each. Each point on these curves represents the mean RMS error values (both actual and filter-computed) over an orbital period plus and minus one standard error. The curve pairs are meant to give an indication of the upper and lower bounds of the σ's during an orbit.

Figures 5-2 illustrate the difference smoothing makes when long times are plotted. An artificial periodic trend is introduced into the plot of the actual RMS error. This is due to averaging over an integer number of observations rather than an exact orbital period. The artificial trend is much smaller in magnitude than the original oscillations so averaging still serves to enhance the legibility of the results.

The previous chapter alluded to the necessity of tuning the estimator with a pseudonoise matrix, CQ_nC^T; indeed, Figures 5-2 were produced with a "tuned" filter. Figures 5-3 illustrate tuning the filter to prevent divergence. In general, this is done by varying CQ_nC^T, running a Monte Carlo simulation, comparing the actual RMS value of the position error to the filter-computed value, and repeating the procedure until the two values are approximately the same *and* the noise is as small as possible such that the filter can run "forever" without diverging (32:337-339). Precise tuning was not attempted in this research; rather, the elements of CQ_nC^T were determined to approximately the correct order of magnitude. [Note, when the predicted

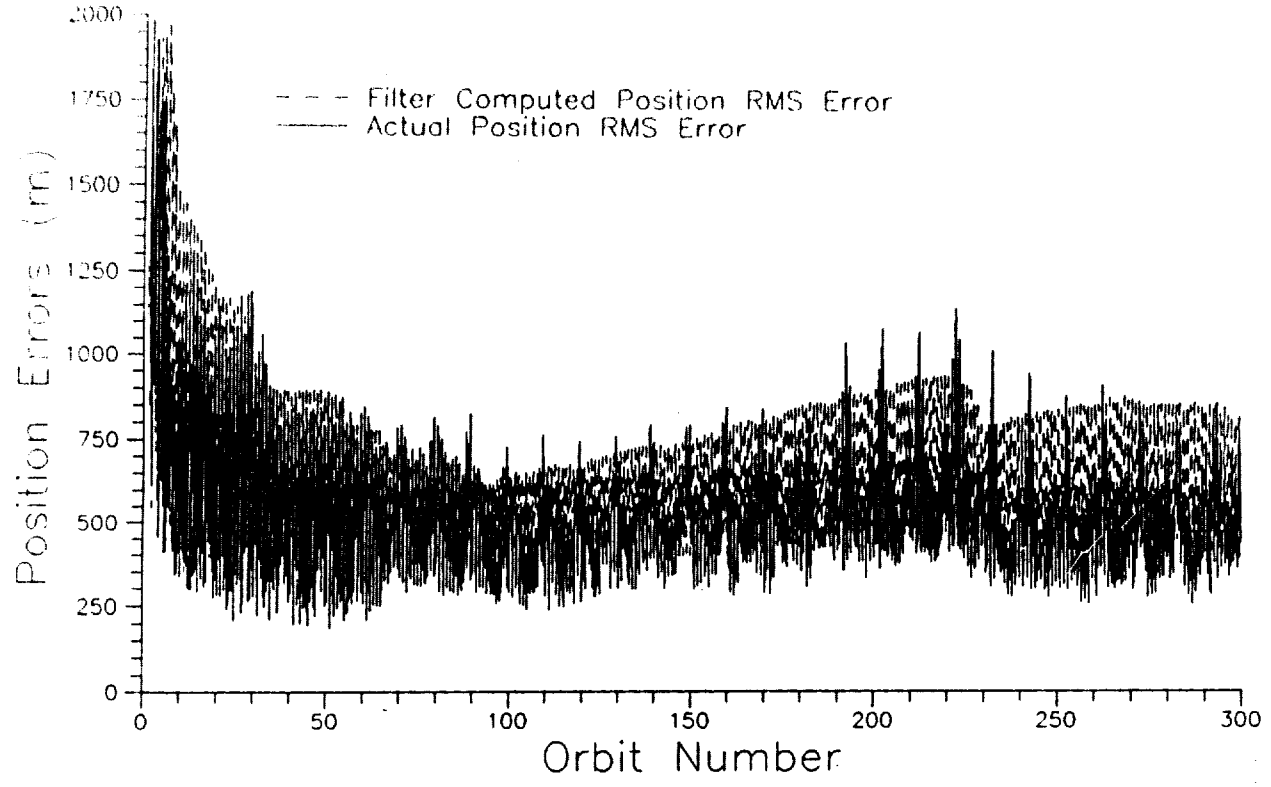

Figure 5-2a Monte Carlo Results for Test Orbit Over Long Times (Unsmoothed)

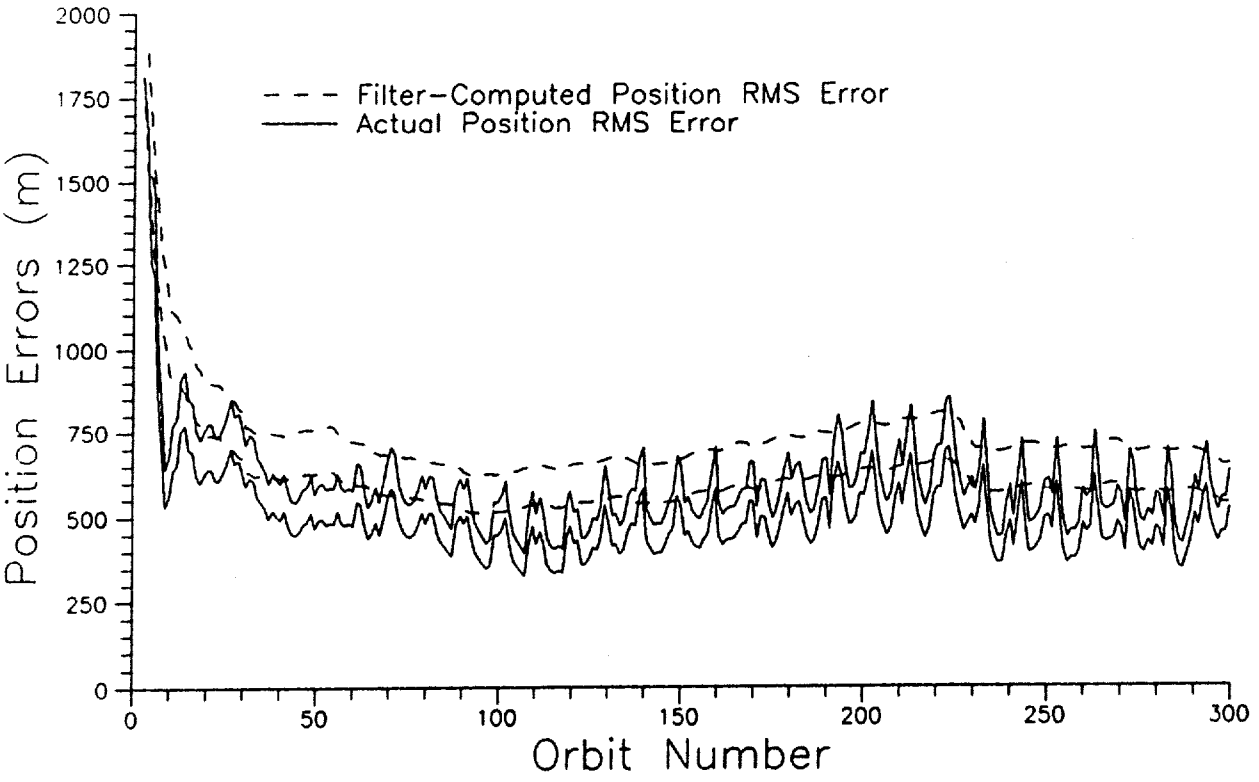

Figure 5-2b Monte Carlo Results for Test Orbit Over Long Times (Smoothed)

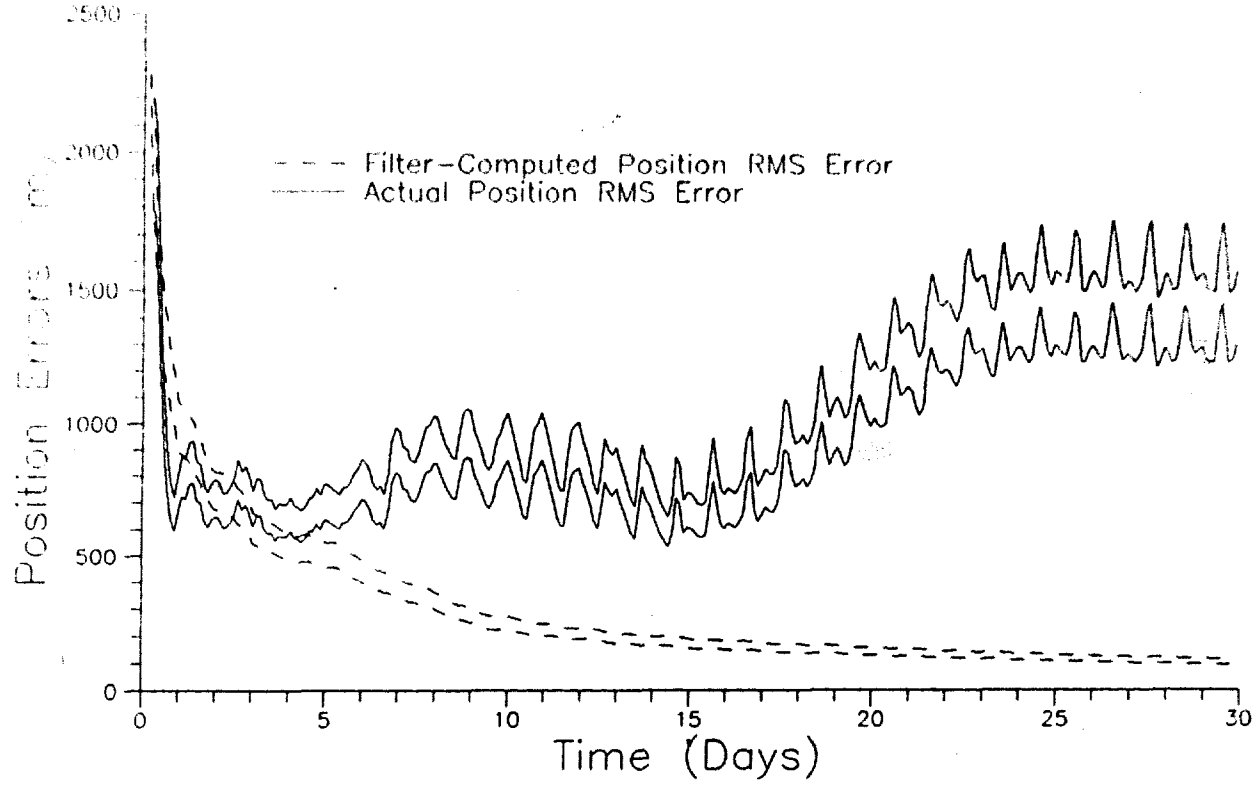

Figure 5-3a Monte Carlo Results for Test Orbit (Untuned Filter)

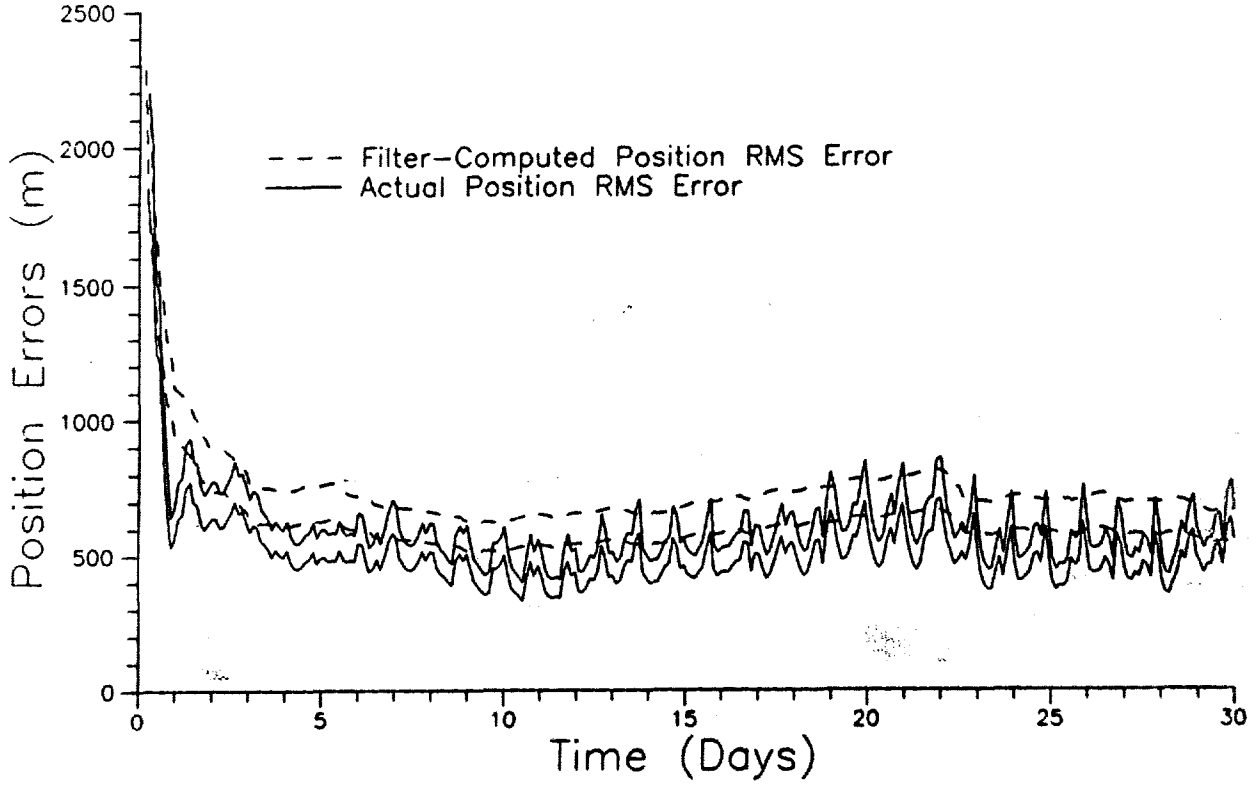

Figure 5-3b Monte Carlo Results for Test Orbit (Properly Tuned Filter)

standard deviation exceeds the actual, then the filter is termed "conservatively tuned," a commonly adopted goal (32:339)] The baseline design was tuned with Q_k = diag $\left(4.1 \times 10^{-13}, 4.3 \times 10^{-13}, 4.1 \times 10^{-10}\ km^2, 1.0 \times 10^{-13}\ rad^2, 1.6 \times 10^{-12}, 7.9 \times 10^{-13}\right)$ and C set equal to the 6 x 6 identity matrix. Figure 5-3b shows that, with this level of pseudonoise, the average steady-state error for the baseline configuration was about 650 m. All forthcoming results were obtained with this pseudonoise matrix unless indicated otherwise.

Variation of Onboard Factors

Certain parameters in the design of this autonomous navigation system can be varied. These include the data observation rate, the precision of (as well as the tolerated bias in) the onboard instrumentation, and the target star sighted to form the angular observation. All of these factors have a direct impact on the accuracy to which a satellite's position can be estimated, so it is appropriate to investigate the effects of varying each factor.

The Effect of Varying the Observation Rate. A satellite may not have an onboard computer dedicated solely to orbit determination; instead, the spacecraft may time-share a computer between navigation, attitude determination, health and maintenance functions, etc. Thus, the tradeoff between computer time allotted to navigation and the precision of position estimates was studied.

As the time between observations increases, one expects the error in the position estimate to grow as well, necessitating the need to retune the filter. If the errors in Aksnes' model are assumed to grow

linearly with time, then an estimate of the error covariance can be obtained by applying the standard conditional expectation operator to the state errors:

$$P_e = E\left(e\ e^T\right) = E\left(\begin{bmatrix} \delta a_f\ \Delta t \\ \delta a_g\ \Delta t \\ \delta a\ \Delta t \\ \delta L\ \Delta t \\ \delta \chi\ \Delta t \\ \delta \psi\ \Delta t \end{bmatrix} \begin{bmatrix} \delta a_f\ \Delta t \\ \delta a_g\ \Delta t \\ \delta a\ \Delta t \\ \delta L\ \Delta t \\ \delta \chi\ \Delta t \\ \delta \psi\ \Delta t \end{bmatrix}^T \right) \tag{5-4}$$

Expanding this out yields:

$$P_e = \begin{bmatrix} \sigma^2_{a_f a_f} & \sigma^2_{a_f a_g} & \sigma^2_{a_f a} & \sigma^2_{a_f L} & \sigma^2_{a_f \chi} & \sigma^2_{a_f \psi} \\ \sigma^2_{a_g a_f} & \sigma^2_{a_g a_g} & \sigma^2_{a_g a} & \sigma^2_{a_g L} & \sigma^2_{a_g \chi} & \sigma^2_{a_g \psi} \\ \sigma^2_{a a_f} & \sigma^2_{a a_g} & \sigma^2_{a a} & \sigma^2_{a L} & \sigma^2_{a \chi} & \sigma^2_{a \psi} \\ \sigma^2_{L a_f} & \sigma^2_{L a_g} & \sigma^2_{L a} & \sigma^2_{L L} & \sigma^2_{L \chi} & \sigma^2_{L \psi} \\ \sigma^2_{\chi a_f} & \sigma^2_{\chi a_g} & \sigma^2_{\chi a} & \sigma^2_{\chi L} & \sigma^2_{\chi \chi} & \sigma^2_{\chi \psi} \\ \sigma^2_{\psi a_f} & \sigma^2_{\psi a_g} & \sigma^2_{\psi a} & \sigma^2_{\psi L} & \sigma^2_{\psi \chi} & \sigma^2_{\psi \psi} \end{bmatrix} \Delta t^2 \tag{5-5}$$

This is a measure of the growth in the covariance between observation times; therefore, it is a *first guess* at predicting the necessary changes in the pseudonoise matrix, CQ_nC^T, when data rates change.

Thus, one would expect the "tuning factors" in CQ_nC^T to increase quadraticly as the time between observations increases.

In the baseline configuration, $Q_n = \text{diag}\left(4.1 \times 10^{-13}, 4.3 \times 10^{-13}, 4.1 \times 10^{-10}\ \text{km}^2, 1.0 \times 10^{-13}\ \text{rad}^2, 1.6 \times 10^{-12}, 7.9 \times 10^{-13}\right)$ and $C = I_{6x6}$ sufficiently tuned the filter to avoid divergence. Eq. (5-5) indicates this should be modified such that

$$Q_n' = Q_n \left(\frac{\Delta t'}{\Delta t}\right)^2 \tag{5-6}$$

when the time between observations changes from the baseline. In terms of conventional units, this implies the pseudonoise matrix for any data rate is $CQ_nC^T = \text{diag}\left(2.3 \times 10^{-15}\ \text{min}^{-2}, 2.4 \times 10^{-15}\ \text{min}^{-2}, 2.3 \times 10^{-12}\ \text{km}^2/\text{min}^2, 5.5 \times 10^{-16}\ \text{rad}^2/\text{min}^2, 8.8 \times 10^{-15}\ \text{min}^{-2}, 4.4 \times 10^{-15}\ \text{min}^{-2}\right)\Delta t^2$ where Δt is the time between observations in minutes.

Figure 5-4 plots the average steady-state position RMS error against the time between observations. To produce this figure, the estimator was retuned in the above manner as Δt was changed from the baseline. Although not perfectly tuned, the estimator does not diverge in the range studied. It does, however, underestimate its own error when $\Delta t < 10$ min. Similarly, for $\Delta t > 10$ min, the filter overestimates its error (conservatively tuned). Notice also that varying the observation rate by a factor of eighty only yields a change in precision of slightly more than a factor of three.

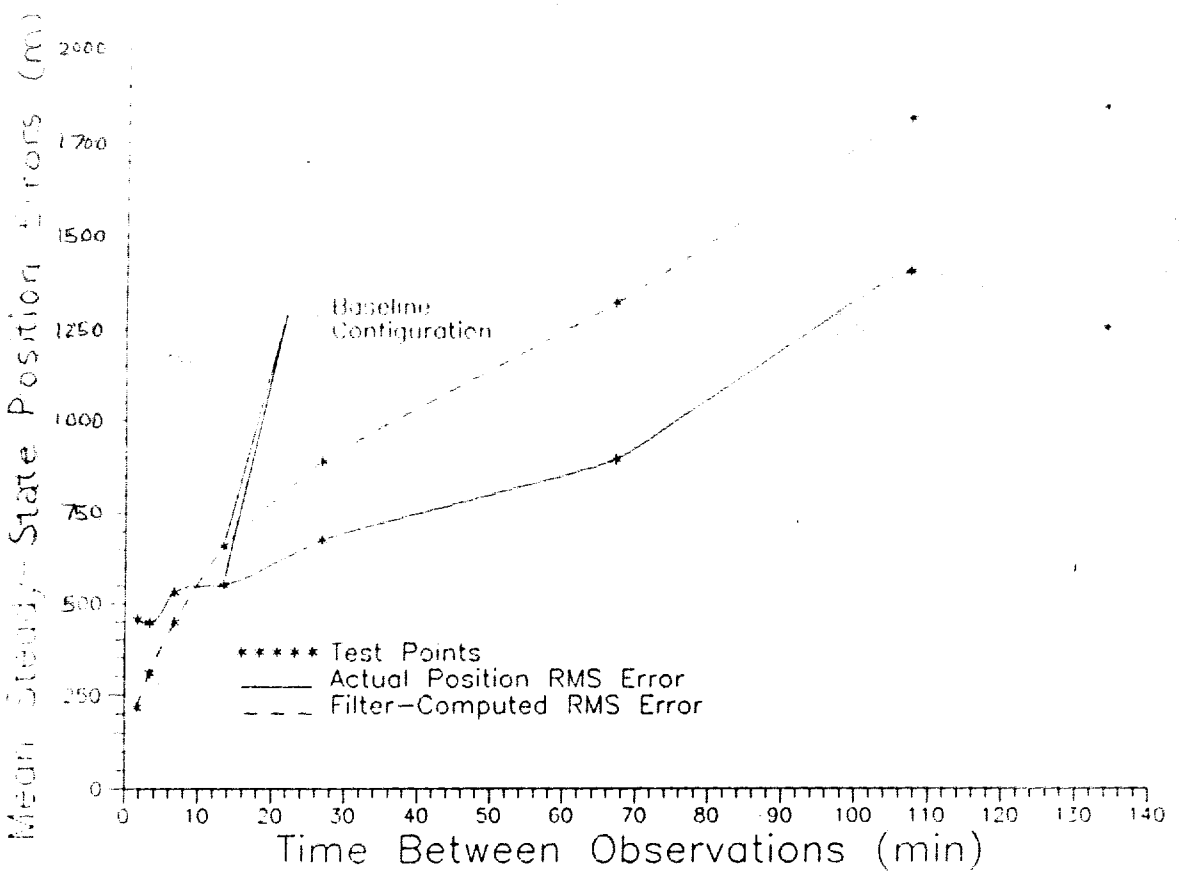

Figure 5-4. The Effect of Data Rate on Steady-State Position RMS Error (Filter Tuned with $CQ_nC^T \propto \Delta t^2$)

In Figure 5-4, the filter over- and underestimates its own error to a large degree. This indicates the assumption $CQ_nC^T \propto \Delta t^2$ is pessimistic. A better assumption is a direct proportionality between time and pseudonoise; i.e., $CQ_nC^T \propto \Delta t$. Results obtained by retuning in this manner are shown in Figure 5-5. Predicted and actual values for the position standard deviation (RMS error) remain at a more constant separation than in Figure 5-4. Further, the values only vary by a factor of 2.7 over the entire range.

Several conclusions can be drawn from these figures. First, the error growth between observation times is not as bad as a linear dependence; rather, it is closer to varying with $\Delta t^{1/2}$. Second, the relative

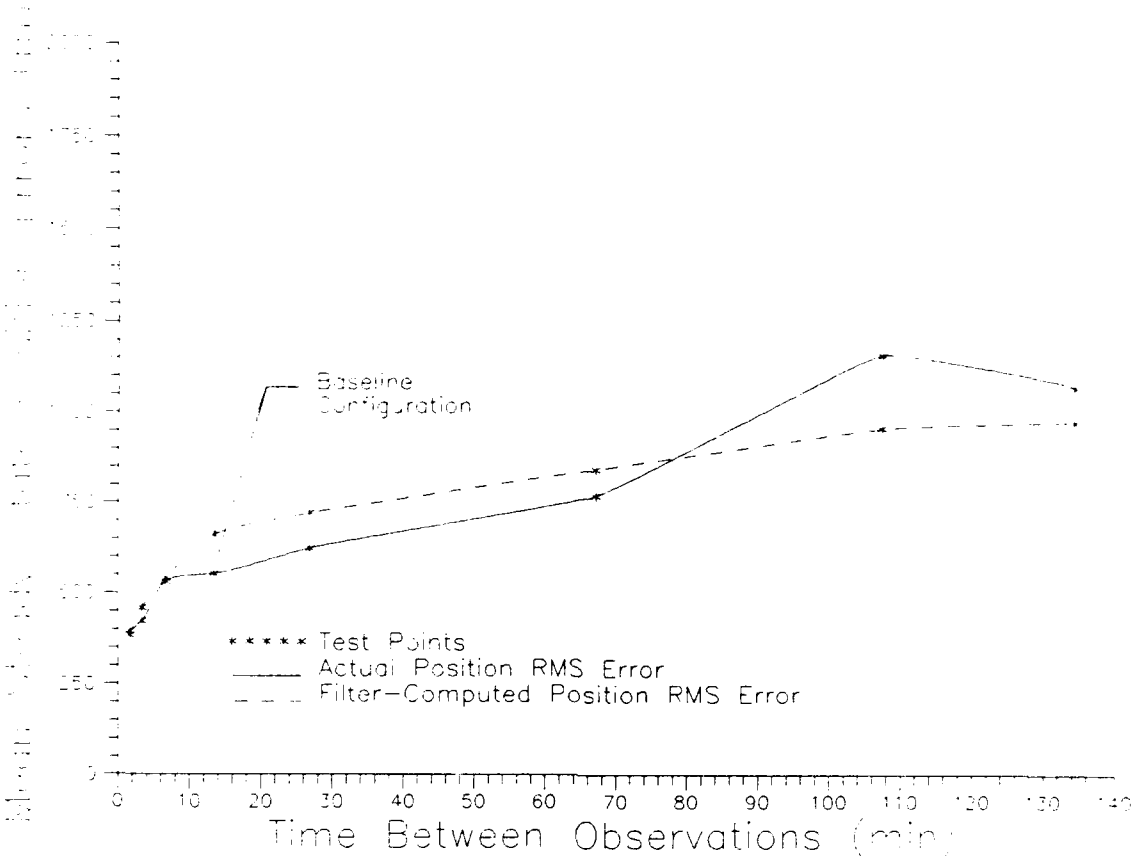

Figure 5-5. The Effect of Data Rate on Steady-State Position RMS Error (Filter Tuned with $CQ_nC^T \propto \Delta t$)

insensitivity of the precision to the data rate allows a wide latitude for time-sharing of computer time. Finally, the error can probably be reduced further with higher data rates, although it should level out as numerical precision limits in Aksnes' model are approached. Studies on an 8 MHz 8088-based microcomputer with an 8087-2 coprocessor indicate that only 5.6 seconds are needed per iteration of the U-D filter. With a maximum of seven iterations allowed, a computer of this class dedicated to navigation is capable of incorporating a new datum every 39 seconds.

The Effect of Instrument Precision. One strength in the system proposed is its use of off-the-shelf hardware. The baseline precision of the star sensor/Earth sensor combination ($\sigma_{comp} = 0.01^{\circ}$) can be obtained with currently available devices while several systems described in Chapter II require much more precise instruments (42:22). Figure 5-6 compares the results obtained by "replacing" the baseline sensor with others of varying precision.

Points representing two other proposed systems, the space sextant and MADAN, are shown in this figure for reference (14:27-32; 17:16; 30:4-5,9). Note, however, this is not an entirely fair comparison. For

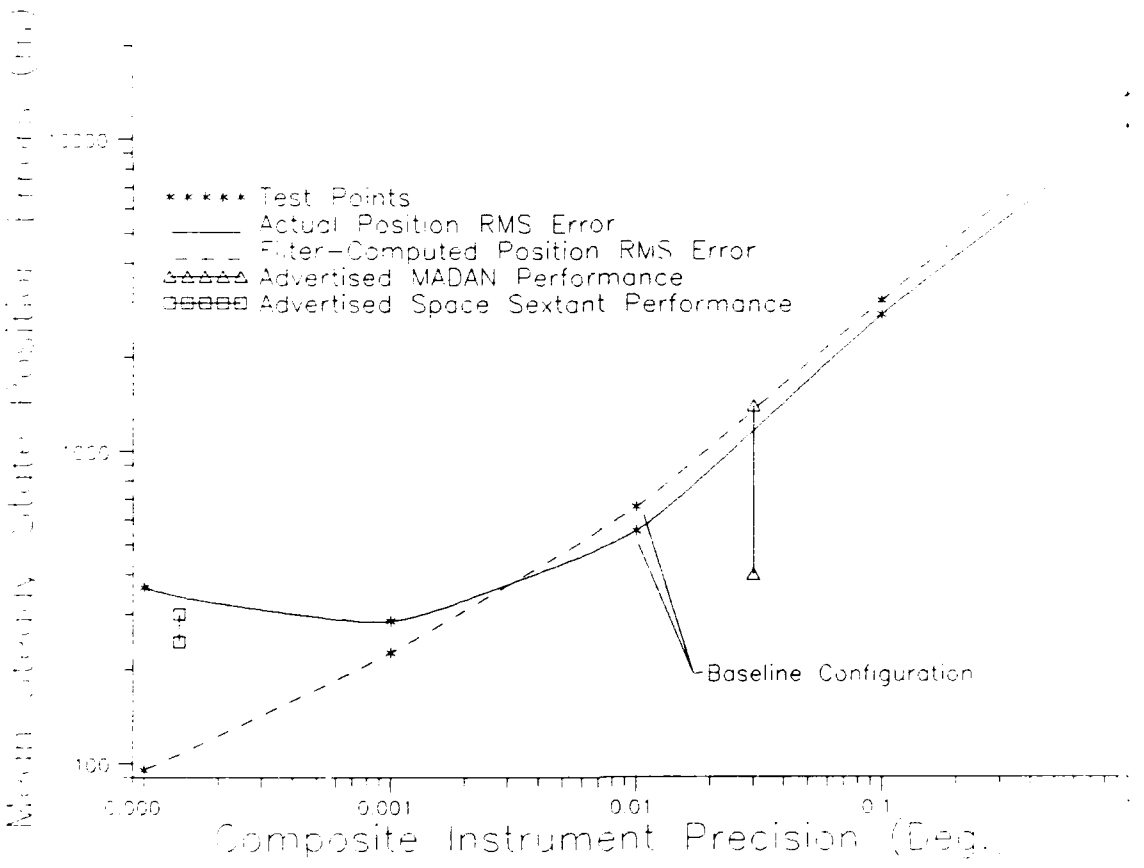

Figure 5-6. The Effect of Composite Instrument Precision on Steady-State Position RMS Error

example, the MADAN system's data rate was 1 Hz compared to the baseline rate of 0.00124 Hz; the observed target was different and the orbits simulated were only approximately the same (14:27-32; 17:16). These numbers are provided only to emphasize that the system put forth in this study compares very favorably with other proposed systems without requiring any new hardware development.

In particular, the space sextant is a proposed system requiring instruments precise to 0.5 arcseconds (0.00014°) to estimate positions to 244-300 m (14:32, 17:16). From the figure, it can be noted that the system addressed in this study is capable of accuracies comparable to the space sextant even when the onboard instrumentation is an order of magnitude *less* precise (0.001°). (Although, in the range $\sigma_{comp} \leq 0.003^{\circ}$ the filter should be retuned, as it underestimates its own error.) Even the baseline configuration is within a factor of two of the space sextant results and approximately the same as the MADAN system.

At this data rate, there is nothing to be gained by employing instruments more precise than 0.001°. Aksnes' theory cannot propagate the trajectory between observations this far apart ($\Delta t \approx 13$ 1/2 min) without losing more precision than is recovered with the next update; i.e., the benefit of the more precise instrument is lost during the state propagation. To take advantage of a better instrument, the data rate must be increased.

The Effect of Instrument Bias. The star sensor/Earth sensor combination employed may not exhibit perfect zero-mean error statistics when measuring the angle γ. Indeed, even if the sensor package displays zero-mean error statistics on the ground, there is a very real possibility

that launch forces will cause the instruments to exhibit a bias when operated in orbit. The question then arises as to how much unmodelled bias the estimator can "absorb" without degrading its performance appreciably.

In creating the simulated data, a constant error, γ_{bias}, can be added to the observed angle, $\hat{\gamma}$, to examine the effect of biased data. This was done and the results are presented in Figure 5-7. The U-D filter was not modified in any way to estimate and/or compensate for biased data. The figure shows that constant measurement errors of up to ±20% of the baseline instrument precision $\left(-0.002^\circ \leq \gamma_{bias} \leq 0.002^\circ\right)$ have virtually no effect on the accuracy of the position estimates.

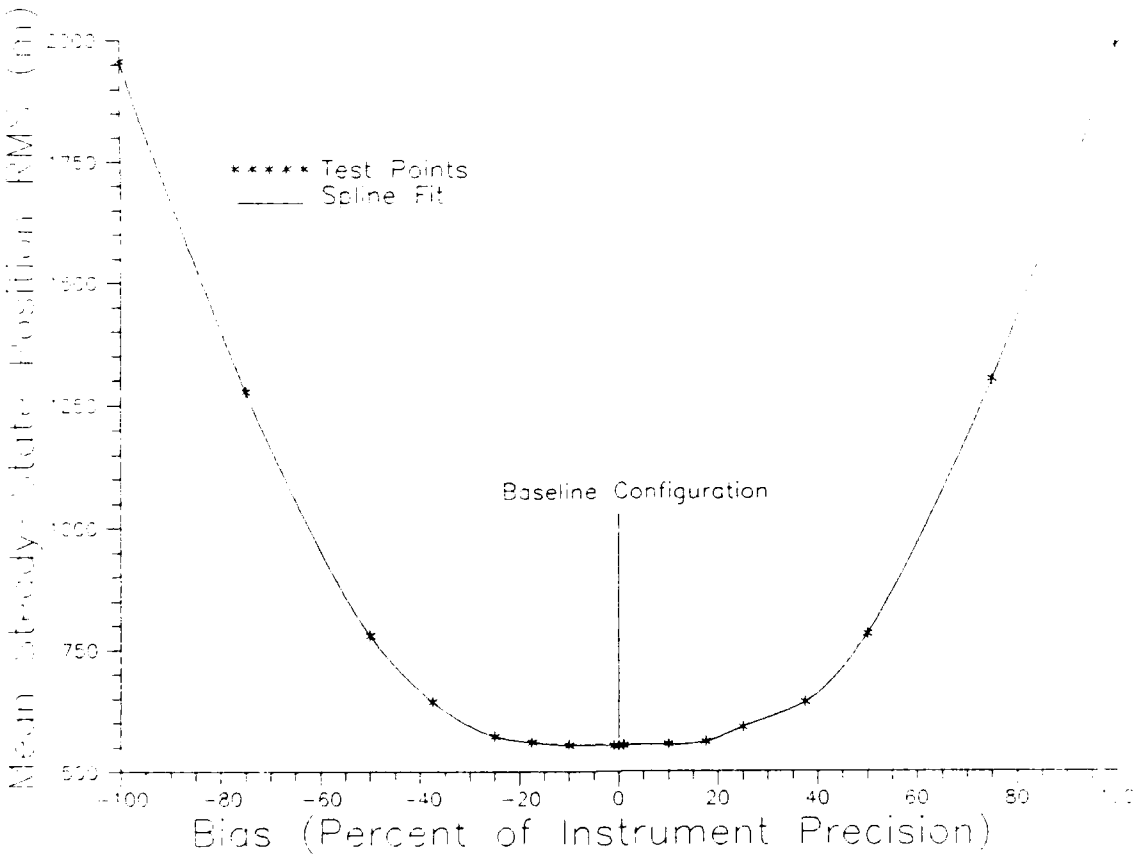

Figure 5-7. The Effect of Instrument Bias on the Steady-State Position RMS Error

Twenty percent of the composite instrument precision represents quite a significant misalignment and steps to avoid errors of this magnitude are taken whenever possible. In the event that the bias falls outside this range, however, it must be estimated and removed from the data. To accomplish this, the filter could be modified to include γ_{bias} as a seventh state to be estimated.

The estimation of the bias can be performed during the spacecraft's initial checkout upon arrival in orbit. Once γ_{bias} is determined, its effect can be removed from every data point prior to updating the state estimate, allowing the filter to resort back to a six-element state vector for day-to-day operation. Aging instruments and other bias-like phenomena may force the bias to be calculated again periodically; however, routine operations of the spacecraft could still only involve a six-element state estimation problem of the type derived in the last chapter. In light of this and the relative insensitivity of the estimator to reasonable levels of bias, a detailed examination is not warranted here.

The Effect of Star Selection. Star selection is not as simple as picking a bright star and observing it. The star must first be in view; i.e., it cannot be hidden behind the Earth. Further, the star must be among those stored in the onboard catalog. The overall goal, however, is to find, identify, and track the visible star which minimizes the error in the position estimate. By looking closely at the geometry of the problem, a few simple a priori statements can be made regarding target stars.

Consider resolving the position error vector into three components along the directions $\hat{e}_r$, $\hat{e}_v$, and $\hat{e}_h$. These unit vectors point along the true radial, velocity, and orbit normal vectors and are referred to as the radial, in-track, and cross-track directions, respectively. Further, recall that noise corrupted samples of the cosine of the angle γ form the data.

If the observed star is parallel to the orbit normal, then $\gamma = \frac{\pi}{2}$ or $\gamma = \frac{3}{2}\pi$ and remains constant regardless of where the satellite is in its orbit; therefore, this arrangement does not yield any information about the in-track error component. Figure 5-8a illustrates this fact. Similarly, an error in the radial direction goes undetected in such an arrangement as is shown in Figure 5-8b. Cross-track errors can be detected; however, and this is presented in Figure 5-8c.

An observed star in the orbit plane, on the other hand, allows the observed angle to cycle through the entire range of values, from 0 to 2π, during each orbit. This maximizes the sensitivity of the observation to in-track and radial motion. Figure 5-9 depicts the observed angle for various spacecraft positions.

Note, when the Earth, spacecraft, and star are aligned ($\gamma = 0$ and $\gamma = \pi$), the observation is completely oblivious to radial errors. While obvious in the situation where the target star is near the orbital plane, this "singularity" is possible any time the three bodies are nearly colinear. If the vehicle lingers in such an arrangement, the error in the radial component of position could grow; therefore, this

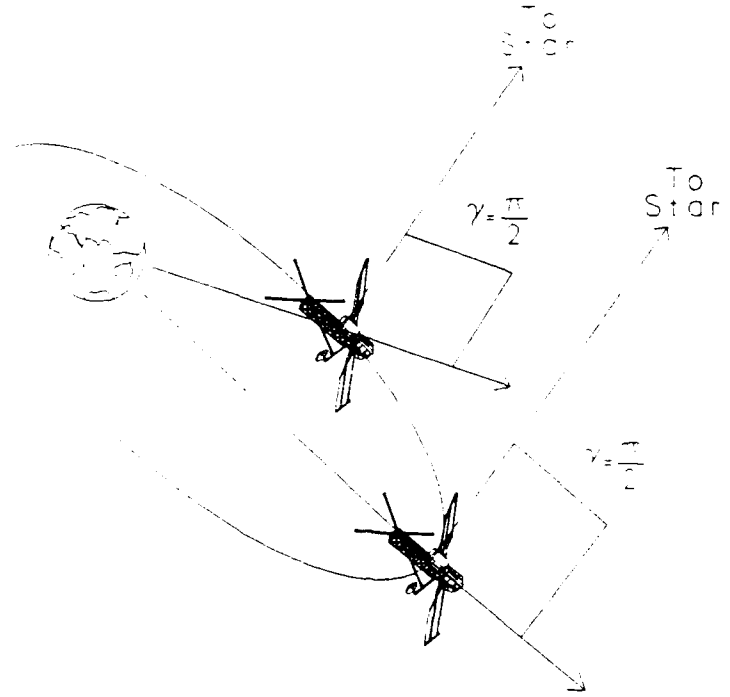

Figure 5-8a Target Star on Orbit Normal
(Ambiguity in In-Track Position)

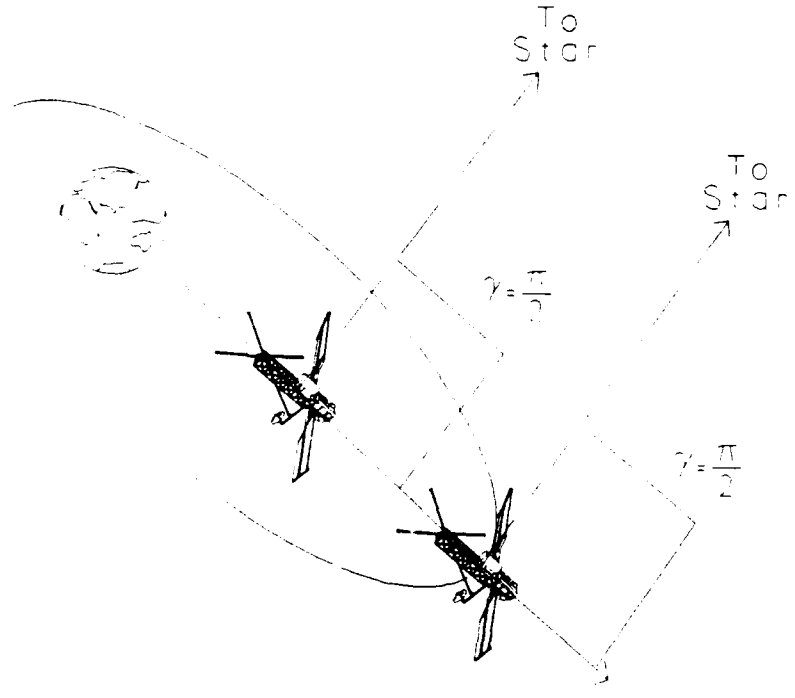

Figure 5-8b Target Star on Orbit Normal
(Ambiguity in Radial Position)

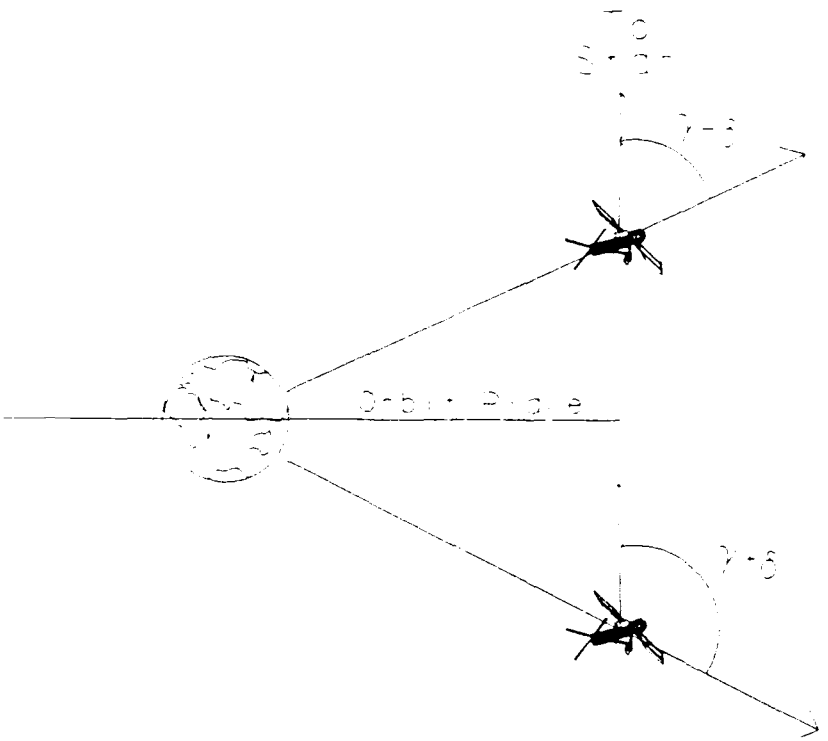

Figure 5-8c Target Star on Orbit Normal
(Ability to Detect Cross-Track Position Errors)

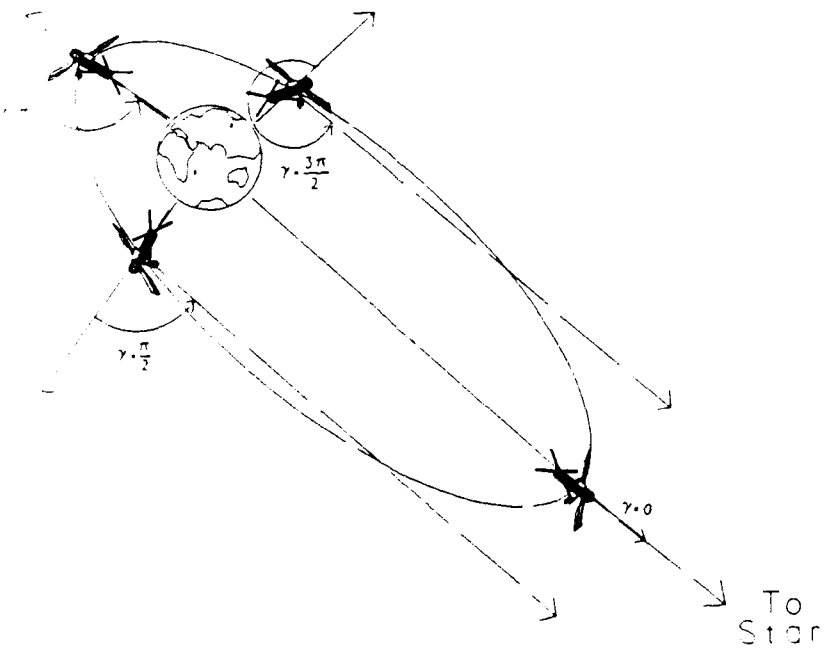

Figure 5-9. Target Star in Orbit Plane
(Variation of γ with In-Track Position)

situation will become of particular concern when synchronous orbits are considered.

The geometric arguments above would seem to indicate a star observed along the orbit normal would maximize the error in the in-track and radial components, while reducing the error in the cross-track component. The goal, however, is to minimize the total error (the magnitude of the distance between the true and predicted positions) and the interplay between components is not intuitively obvious. Monte Carlo studies are useful to investigate this interaction.

Figure 5-10 presents the steady-state RMS error resolved into components along $\hat{e}_r$, $\hat{e}_v$, and $\hat{e}_h$ for various target stars. The small onboard star catalog implies that there are various times when known stars did not exist *exactly* at the target angles shown. Instead, the true star observed was the visible star closest to the target angle. To emphasize the trends for analysis and remove the effects of choosing the "next best" star, quadratic fits are shown in the figure. These curves fit the data quite well, especially in the region $\theta_{target} \leq 30^{\circ}$.

The figure clearly shows that the a priori insights into the behavior of all three components of error were correct. Overall, the total error is a minimum (~460 m) when sighting stars approximately 22° out of the orbit plane. In summary, proper star selection is quite important, as it can mean a factor of three reduction in error. Notice further, the filter is well tuned initially and remains well-tuned throughout a wide range of target angles.

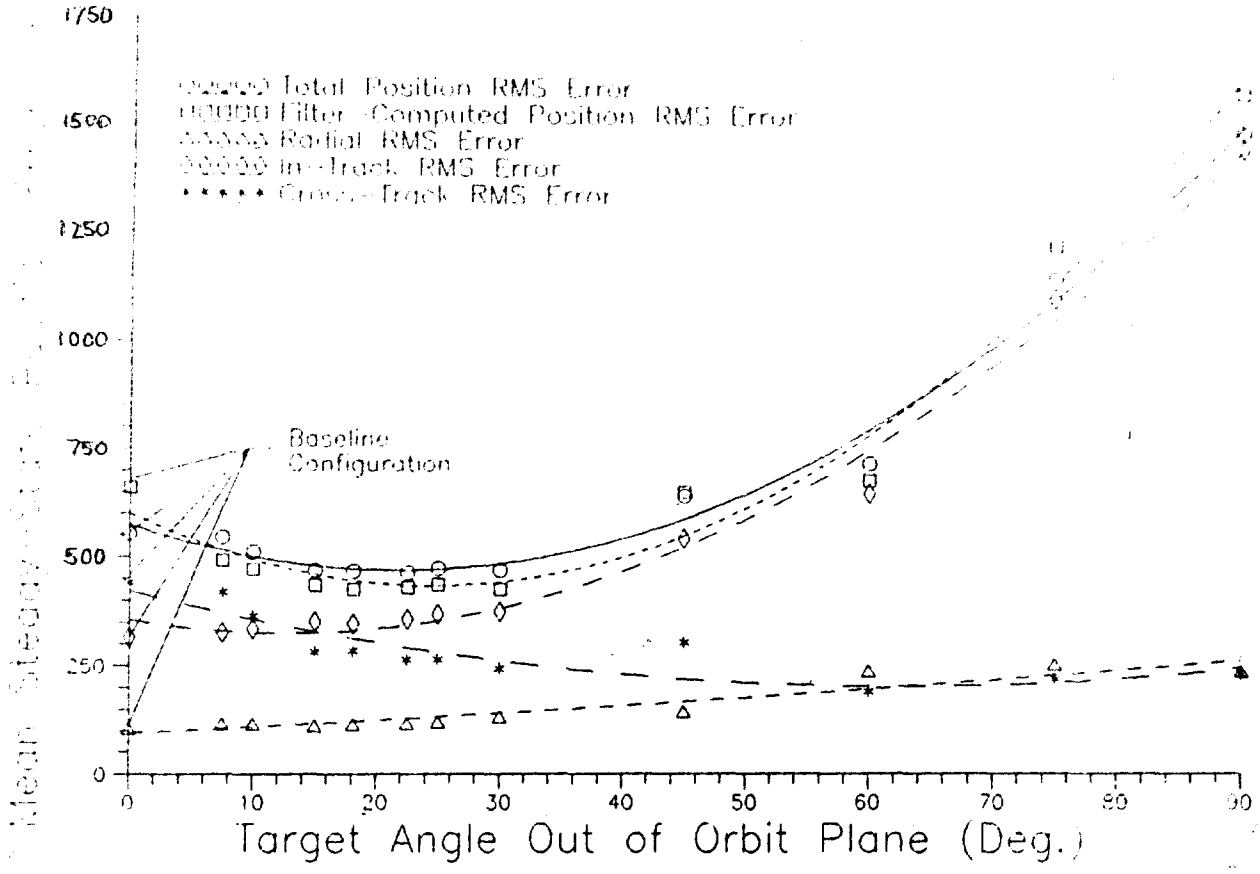

Figure 5-10. The Effect of the Observed Star on Steady-State Position RMS Error

For completeness, it should be noted a Monte Carlo simulation in which two stars were alternately observed in a "round-robin" fashion was performed. RMS error values for this case were approximately the mean between those resulting from observing either star exclusively. Quite simply, in this orbit there is no advantage in sighting more than one θ_{target}. Results in a later section will show an exception to this in the case of high-altitude orbit.

Battin has put forth another method of star selection based on the covariance estimate $P(t_1^-)$ (8:687-690). Simply put, his method selects

the available observation star which results in the maximum reduction in the 1σ position error sphere at time t_i. When applied to this configuration, a Monte Carlo study indicated the steady-state RMS value of position error was 452 m. The increase in precision with this method over a single optimum star angle was very modest (-8 m) for the extra computations involved. Further, actual employment of this method would require satellites to reorient star trackers rapidly before every observation. For these reasons, Battin's method was abandoned and not applied to any further situations.

Assembling the Best Practical Configuration. Results presented in the last few sections allow the user to minimize the error in position estimates by varying one parameter at a time. Observations spaced as close together as possible appeared to be the best data rate. The star sensor/Earth sensor combination was found to produce the smallest RMS error values with a precision of 0.001°. Finally, the selection of which star to be observed was shown to have quite an effect on the accuracy of this system. In particular, it was shown that stars near 22° out of the orbit plane produced the best results for *this particular orbit*. The question arises as to whether these findings can be combined into a realistic system producing better results than were obtained by varying the parameters individually.

If it is assumed navigation, attitude determination, and health and maintenance functions share an onboard computer equally, then approximately a third of computer's time can be spent computing the position. Recall, the results cited for this algorithm on an 8088-based computer stated a maximum of 39 seconds would be required for this function.

Most data points would require less than this, so an observation every 100 seconds (0.01 Hz) is reasonable to assume. Scaling the pseudonoise from the baseline rate such that $CQ_nC^T \propto \Delta t$ produces the matrix $CQ_nC^T = \text{diag}\left(5.1 \times 10^{-14},\ 5.4 \times 10^{-14},\ 5.1 \times 10^{-11}\ \text{km}^2,\right.$ $\left.1.3 \times 10^{-14}\ \text{rad}^2,\ 2.0 \times 10^{-13},\ 9.9 \times 10^{-14}\right)$. Composite instrument precisions are limited by *currently* available Earth sensors to approximately 0.01°; therefore, a system utilizing only off-the-shelf hardware would be limited to $\sigma_{comp} = 0.01^\circ$ (42:22). In light of Figure 5-7, any reasonable instrument bias can be assumed to be zero. Finally, with proper instrument placement, a star sensor can lock onto and track any bright star in view that is in the onboard catalog, so it is a trivial problem to sight a star near the optimum target angle.

Figure 5-11 illustrates the performance of a system based on a 0.01 Hz data rate, $\sigma_{comp} = 0.01^\circ$. and a target angle of 22.5° out of the orbit plane. The mean steady-state RMS error is 337 m. The filter slightly underestimates its own error at 291 m due to a minor mismatch in the tuning, but the performance is, overall, excellent and the filter does not diverge. Recall that the space sextant's advertised precision is 224 - 300 m and the value of this system to orbit determination is apparent.

Variation of the Orbital Elements

The last chapter introduced the equinoctial elements as an attempt to avoid problems associated with singularities in the classical orbital elements, particularly those at $e = 0$ and $i = 0$. Since many useful orbits involve one or both of these situations, an examination into the

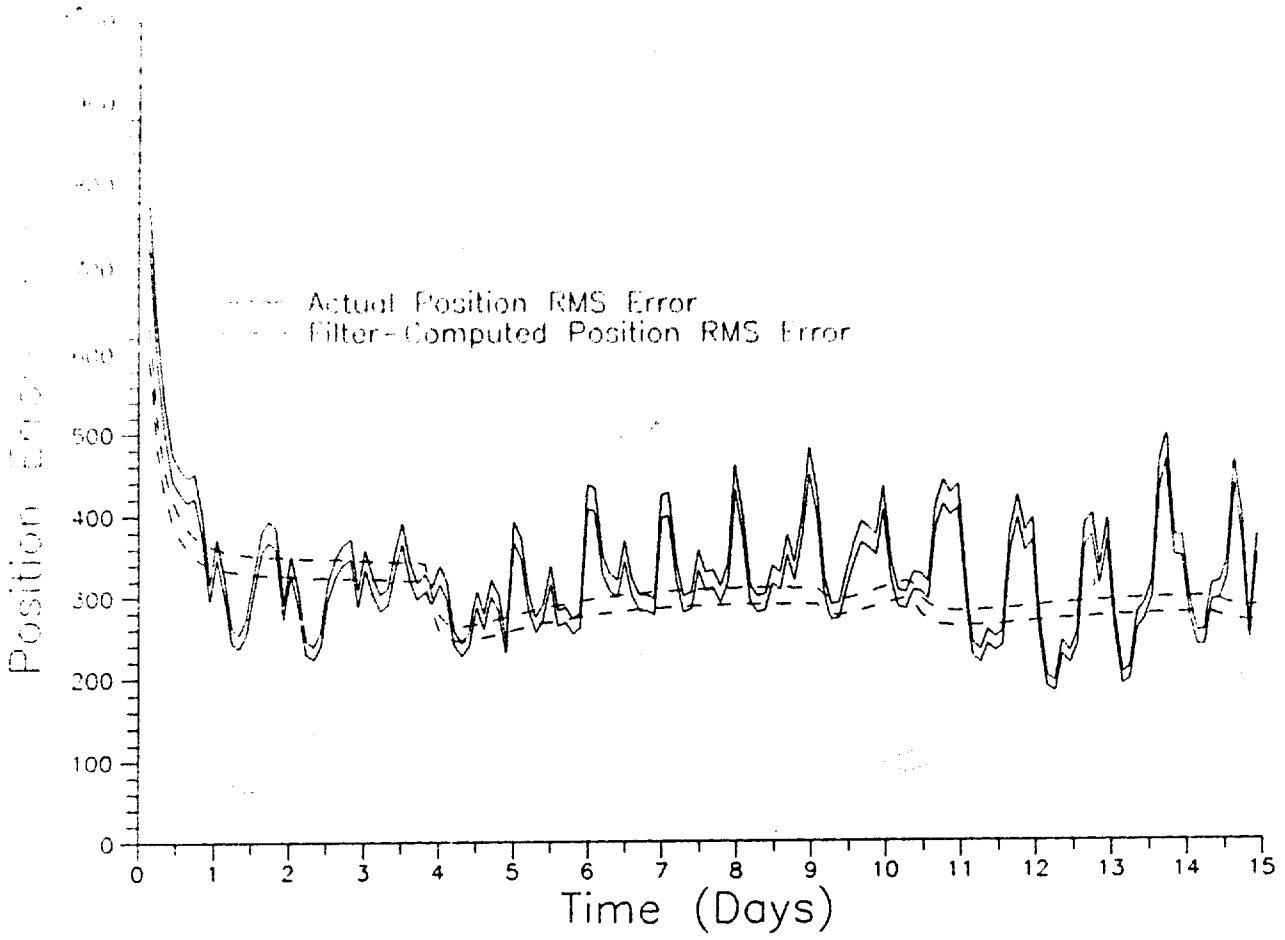

Figure 5-11. Monte Carlo Results for Best Practical Configuration

behavior of this system near these points is warranted. For example, low-Earth orbits (LEO) can be expected to approach $e = 0$ with time, so it is particularly useful to examine how the filter behaves as the eccentricity tends towards zero (25:8,143).

If the equinoctial elements are written in terms of the classical elements

$$a_g = e \cos\pi \tag{5-7a}$$

$$a_g = e \sin\pi \tag{5-7b}$$

$$a = a \tag{5-7c}$$

$$L = \pi + M \tag{5-7d}$$

$$\chi = \frac{\sin i \sin\Omega}{1 + \cos i} \tag{5-7e}$$

$$\psi = \frac{\sin i \cos\Omega}{1 + \cos i} \tag{5-7f}$$

where $\pi = \omega + \Omega$ is the longitude of perigee, then the variation can be taken:

$$\delta a_f = \delta e \cos\pi - e \sin\pi \, \delta\pi \tag{5-8a}$$

$$\delta a_g = \delta e \sin\pi + e \cos\pi \, \delta\pi \tag{5-8b}$$

$$\delta a = \delta a \tag{5-8c}$$

$$\delta L = \delta(\pi + M) \tag{5-8d}$$

$$\delta\chi = \frac{\sin i \cos\Omega}{1 + \cos i} \delta\Omega + \frac{\sin\Omega}{1 + \cos i} \delta i \tag{5-8e}$$

$$\delta\psi = - \frac{\sin i \sin\Omega}{1 + \cos i} \delta\Omega + \frac{\cos\Omega}{1 + \cos i} \delta i \tag{5-8e}$$

Squaring and applying the standard conditional expectation operator to both sides yields

$$\sigma^2_{a_f a_f} = \cos^2\pi \, \sigma^2_{ee} + e^2 \sin^2\pi \, \sigma^2_{\pi\pi} \tag{5-9a}$$

$$\sigma^2_{a_g a_g} = \sin^2\pi \, \sigma^2_{ee} + e^2 \cos^2\pi \, \sigma^2_{\pi\pi} \tag{5-9b}$$

$$\sigma^2_{aa} = \sigma^2_{aa} \tag{5-9c}$$

$$\sigma^2_{LL} = \sigma^2_{(\pi+M)(\pi+M)} \tag{5-9d}$$

$$\sigma^2_{\chi\chi} = \left(\frac{\sin i \cos\Omega}{1 + \cos i}\right)^2 \sigma^2_{\Omega\Omega} + \left(\frac{\sin\Omega}{1 + \cos i}\right)^2 \sigma^2_{ii} \tag{5-9e}$$

$$\sigma^2_{\psi\psi} = \left(\frac{\sin i \sin\Omega}{1 + \cos i}\right)^2 \sigma^2_{\Omega\Omega} + \left(\frac{\cos\Omega}{1 + \cos i}\right)^2 \sigma^2_{ii} \tag{5-9f}$$

where the cross terms have been ignored as a rough approximation. Near the baseline orbit, the argument of perigee is $\omega = 40^\circ$ and the longitude of the ascending node is $\Omega = 50^\circ$. Substituting these values into Eqs. (5-9) simplifies the expressions:

$$\sigma^2_{a_f a_f} = e^2 \sigma^2_{\pi\pi} \tag{5-10a}$$

$$\sigma^2_{a_g a_g} = \sigma^2_{ee} \tag{5-10b}$$

$$\sigma^2_{aa} = \sigma^2_{aa} \tag{5-10c}$$

$$\sigma^2_{LL} = \sigma^2_{(\pi+M)(\pi+M)} \tag{5-10d}$$

$$\sigma^2_{\chi\chi} = \left(\frac{\sin i \cos 50^\circ}{1 + \cos i}\right)^2 \sigma^2_{\Omega\Omega} + \left(\frac{\sin 50^\circ}{1 + \cos i}\right)^2 \sigma^2_{ii} \tag{5-10e}$$

$$\sigma^2_{\psi\psi} = \left(\frac{\sin i \sin 50^\circ}{1 + \cos i}\right)^2 \sigma^2_{\Omega\Omega} + \left(\frac{\cos 50^\circ}{1 + \cos i}\right)^2 \sigma^2_{ii} \tag{5-10f}$$

These expressions represent the uncertainty in the corresponding orbital elements. Significant growth in any of these terms indicates an ill-defined element and will be reflected as larger errors in state estimates. The total position error reflects a combination of errors in all six elements and a significant error in any one element will be reflected as a change in this total; therefore, the total is a more important parameter than any individual component. It is sufficient (for this study) to monitor the RMS error values for signs of a breakdown in the system as the eccentricity and inclination are independently driven to zero.

Driving the Eccentricity to Zero. As the eccentricity approaches zero, the argument of the perigee, ω, becomes undefined as does the longitude of perigee, π. The mean longitude, $L = \pi + M$, remains well-defined in spite of the uncertainty of ω. The uncertainty in π will be reflected in the growth of $\sigma^2_{\pi\pi}$. Thus, the only term in Eqs. (5-10) that could experience significant growth is $\sigma^2_{a_f a_f}$ and then only if $\sigma_{\pi\pi}$ goes to infinity faster than e goes to zero.

Another consideration involved when comparing different eccentricities is the effect the target angle has on the precision of position estimates. In other words, can a fair comparison of the RMS errors be made by varying the eccentricity while holding the target star angle constant? Figure 5-12 plots the steady-state RMS value of position error against the target star angle for the test orbit with varying eccentricities. Quadratic approximations have been fit to the data once again to "smooth out" the effects of a small onboard catalog and enhance

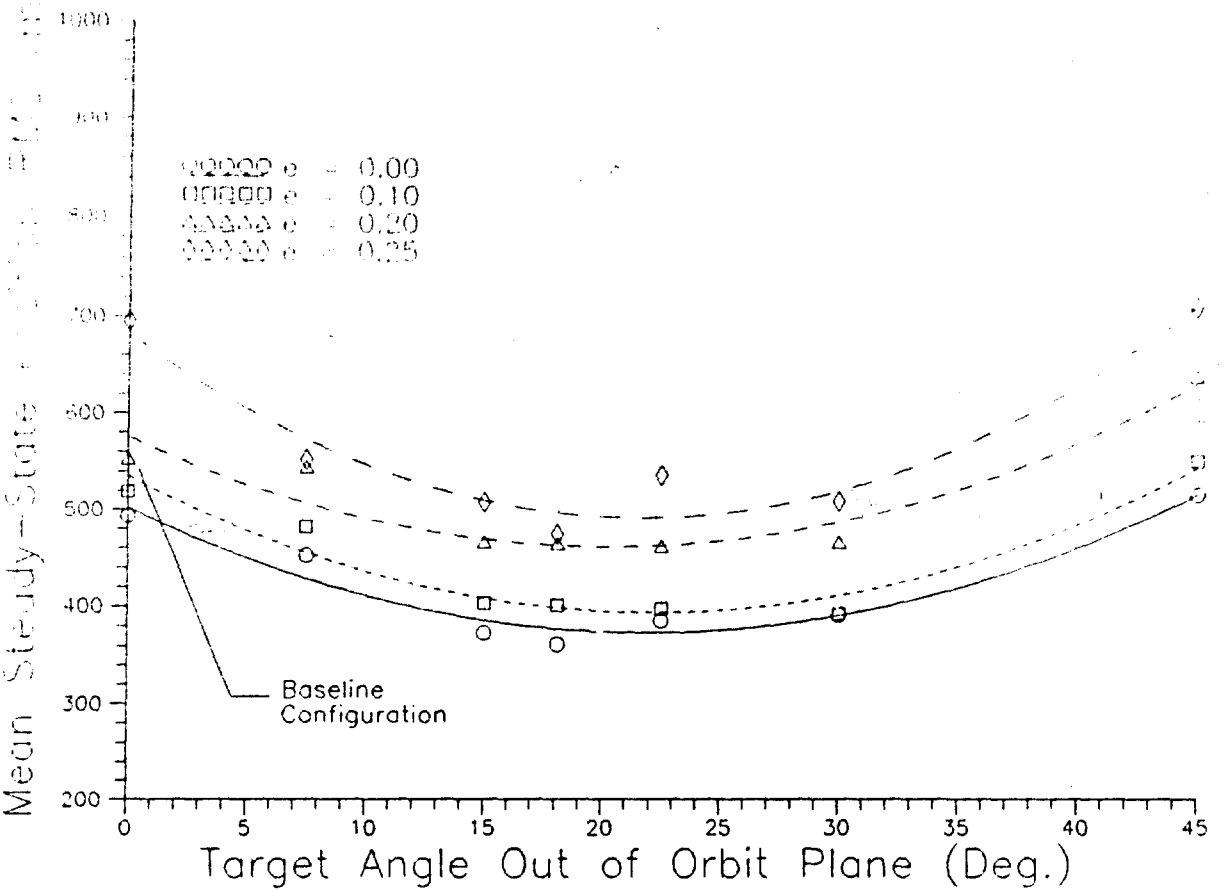

Figure 5-12. The Effect of the Observed Star on Orbits of Various Eccentricities

the viewing of overall trends. Note that the curves for each value of e maintain their relative spacing for all target angles; therefore, a fair comparison between orbital eccentricities can be made when the observation star is held constant at any angle. This also indicates that the optimum target angle would not have to be reprogrammed in the satellite if the eccentricity changes due to drag, etc.

Figure 5-13 contains many more data points than the previous figure. The additional data and apparent three-dimensionality helps reveal trends

better. Shown clearly in the figure, the optimum star for observation remains between 15° and 25° out of the orbit plane, regardless of the eccentricity. Further, Figure 5-13 shows RMS errors generally increasing with eccentricity.

Having determined that the target star can be held constant ($\theta_{target} = 0°$) while varying the eccentricity, Figure 5-14 shows the steady-state components of the position error for $0 \leq e \leq 0.25$. The total position error decreases as $e \Rightarrow 0$, indicating that none of the variances in Eqs. (5-10) increase significantly. Simply put, the use of the equinoctial elements allows the filter to perform equally well (without retuning) over a wide range of values for e. The classical elements could not be expected to exhibit such robustness due to singularities.

Driving the Inclination to Zero. As the orbital inclination approaches zero, the longitude of the ascending node, Ω, becomes undefined. All other classical elements remain well-defined, so their variances should not change significantly. Thus, only the terms $\sigma^2_{\chi\chi}$ and $\sigma^2_{\psi\psi}$ in Eqs. (5-10) appear to have the possibility of growing as $i \Rightarrow 0$.

As in the study of $e \Rightarrow 0$, it is helpful to plot RMS values of the total error for various inclinations against a range of target star angles. This is presented in Figure 5-15 with quadratic approximations to the data. While the relative spacing changes somewhat over the range of θ_{target}, there is enough consistency at the low end for a fair comparison between various orbital inclinations while holding the observation star constant at $\theta_{target} = 0°$. A larger number of data points for

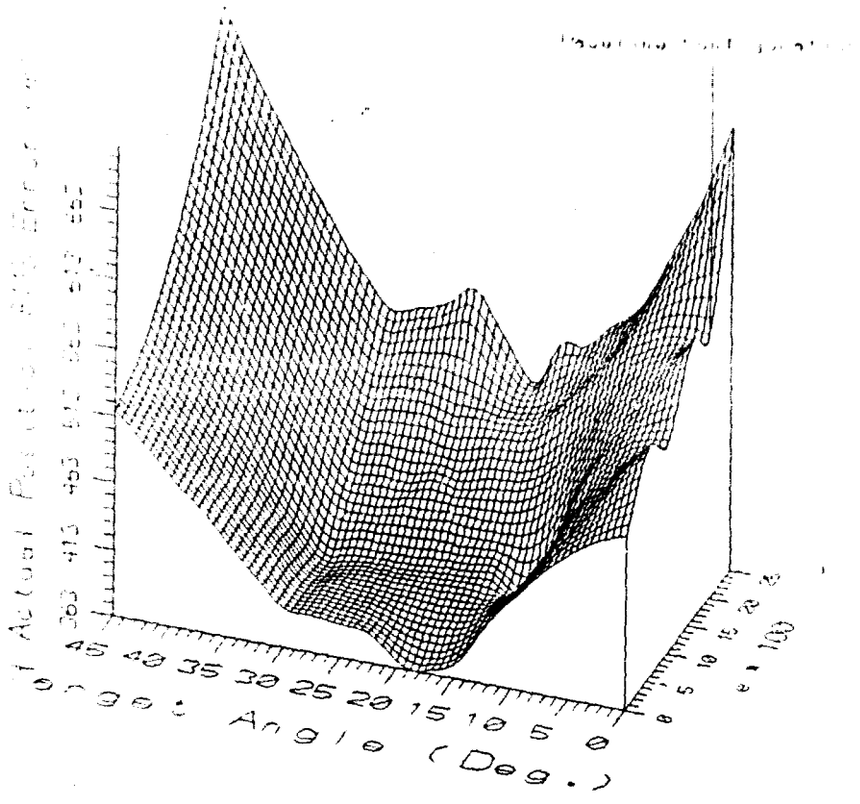

Figure 5-13. Steady-State Results for $0^\circ \leq \theta_{target} \leq 45^\circ$ and $0 \leq e \leq 0.25$

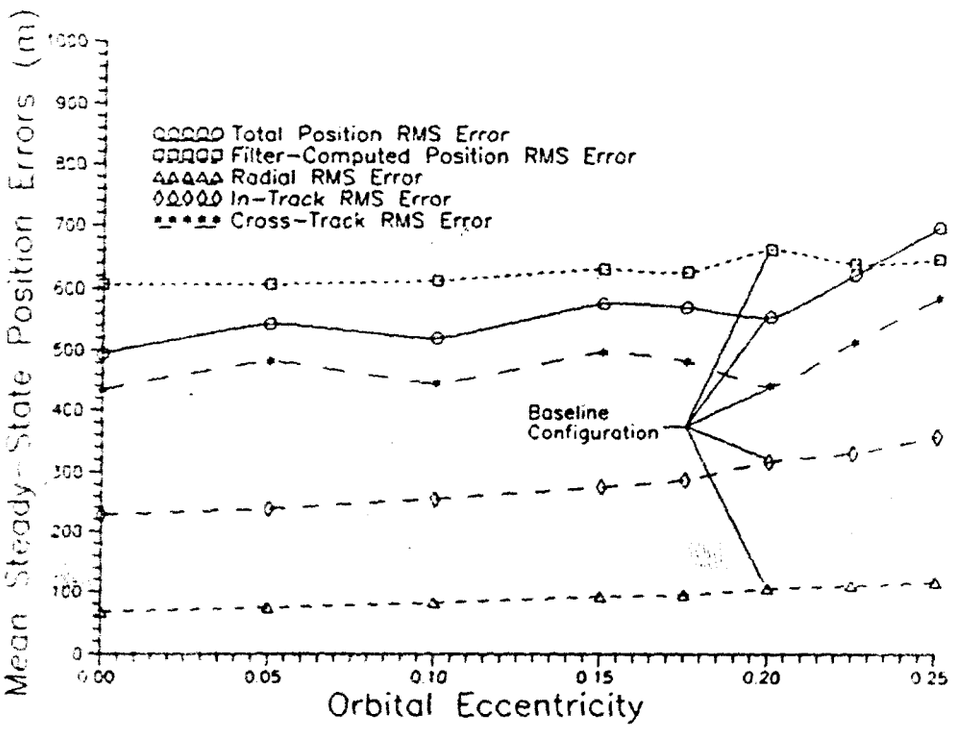

Figure 5-14. The Effect of Varying Eccentricity on Steady-State Position RMS Error ($0 \leq e \leq 0.25$)

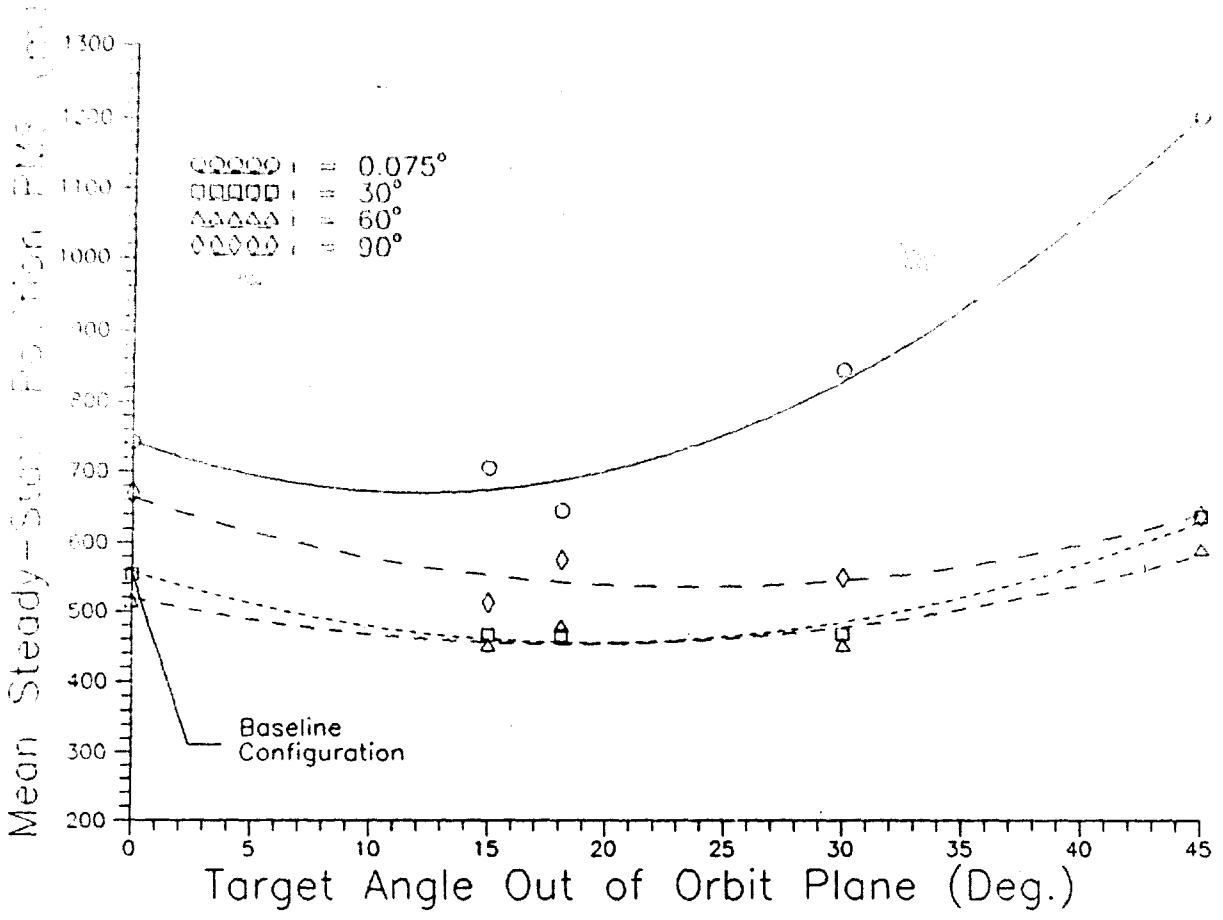

Figure 5-15. The Effect of the Observed Star on Orbits of Various Inclinations

inclinations from 0.075° to 90° is presented in Figure 5-16 to illustrate trends not easily represented in line graphs such Figure 5-15. Once again, the optimum target angle lies near 20° for all inclinations.

The effect of varying the inclination between 0.075° and 90° in the baseline configuration is shown in Figure 5-17. Over this range the filter rapidly converges to steady-state conditions. Note, however, the filter underestimates its error outside the range $15° \leq i \leq 45°$, implying the tuning is more sensitive to changes in inclination than it

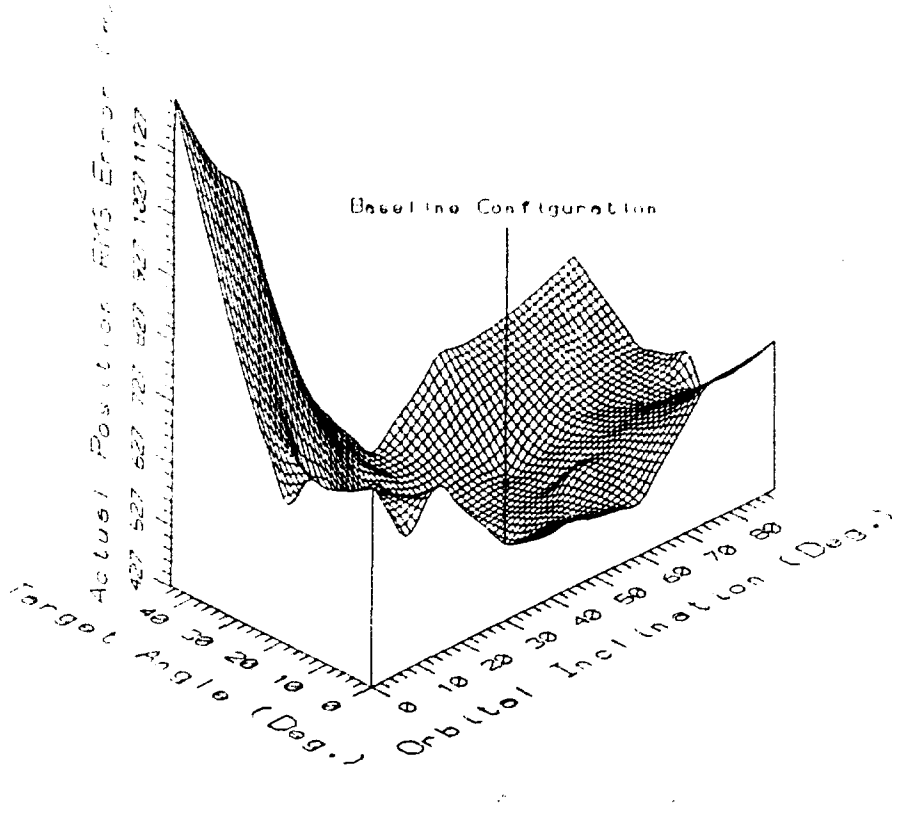

Figure 5-16. Steady-State Results for $0^\circ \leq \theta_{target} \leq 45^\circ$ and $0.075^\circ \leq i \leq 90^\circ$

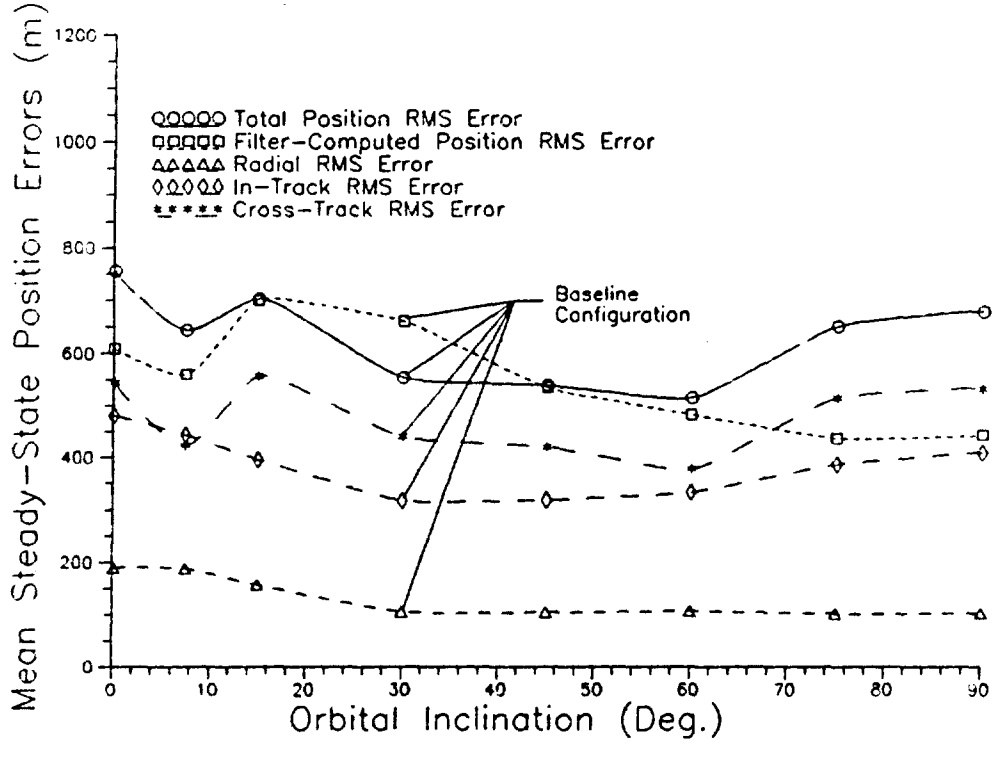

Figure 5-17. The Effect of Varying Inclination on Steady-State Position RMS Error ($0.075^\circ \leq i \leq 90^\circ$)

was to changes in eccentricity. RMS values of the position error, both actual and predicted, grow as the inclination is reduced, indicating a growth in the uncertainty of one or more orbital elements.

Figure 5-18 depicts the situation at smaller inclinations. When i becomes smaller than 0.075°, the filter no longer converges to steady-state conditions; rather, the errors continue to grow with time. For illustrative purposes only, the values shown in this figure are simply one-month averaged errors. The in-track and cross-track components of the error show the largest degree of divergence.

In zero-inclination situations, the mean longitude itself, $L = \omega + \Omega + M$, is well-defined; however, when explicitly constructed from the individual components ω, Ω, and M, the mean longitude can become somewhat uncertain. Since L directly relates to the in-track position, the growth in in-track error can be explained by the uncertainty involved in constructing L. The cross-track motion is, on the other hand, directly related to the uncertainty in the orbital inclination. The inclination is derived from χ and ψ, the two elements whose uncertainty can be expected to become large as $i \Rightarrow 0$. Thus, the growth in cross-track error can be explained also.

The divergent situation when $i \Rightarrow 0$ does not imply the change to equinoctial elements was in vain. Indeed, simply retuning the filter retrieves the convergence. Setting $CQ_nC^T = \text{diag}\left(4.1 \times 10^{-13}, 4.3 \times 10^{-13}, 4.1 \times 10^{-3}\ \text{km}^2, 1.0 \times 10^{-13}\ \text{rad}^2, 1.6 \times 10^{-12}, 7.9 \times 10^{-13}\right)$ by trial and error results in steady-state actual and filter-computed

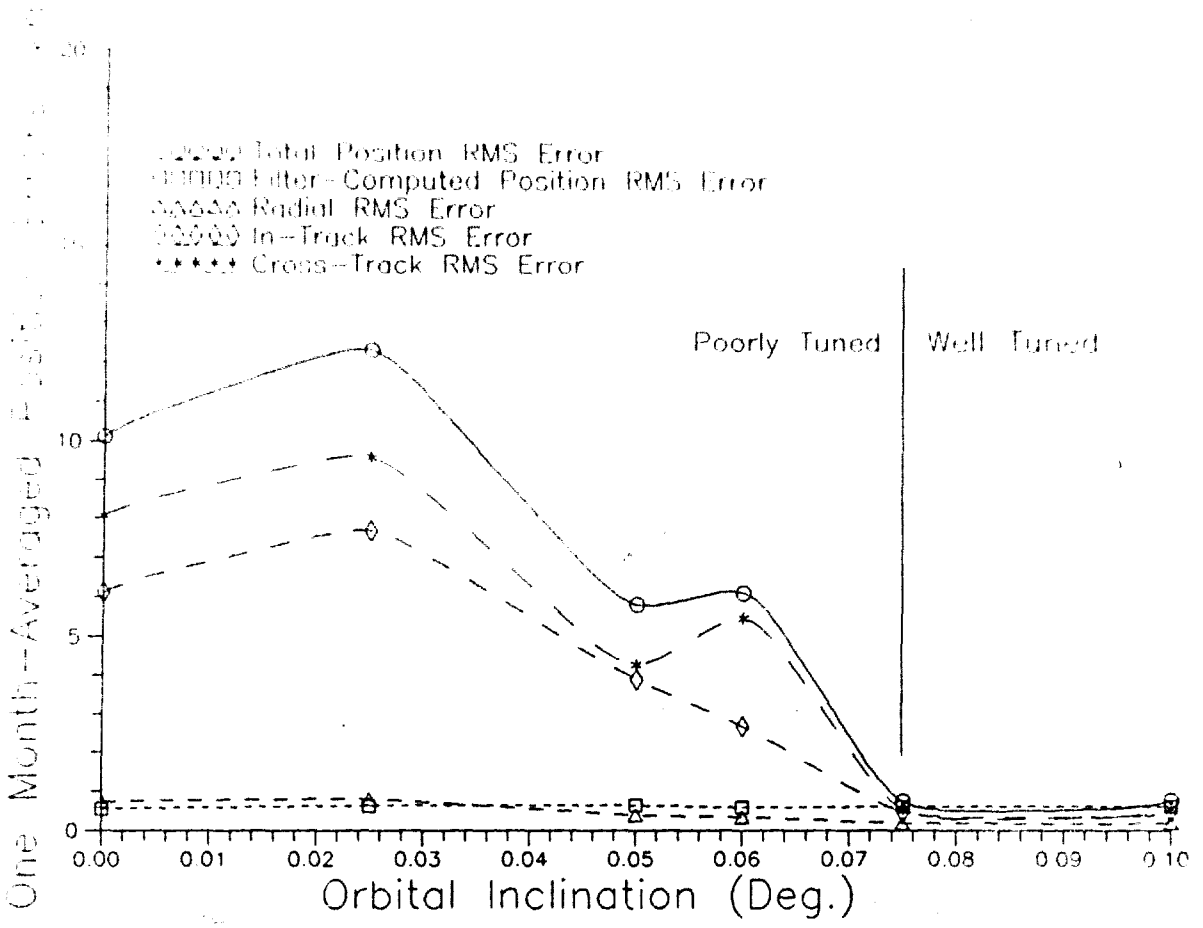

Figure 5-18. The Effect of Driving the Inclination to Zero (Without Retuning the Filter)

standard deviation values of approximately 1 km each, as shown in Figure 5-19. The only element given more pseudonoise was that one corresponding to the semimajor axis. This allowed the iteration scheme in the U-D filter more latitude, with the result that this filter could, indeed, converge for *all* inclinations.

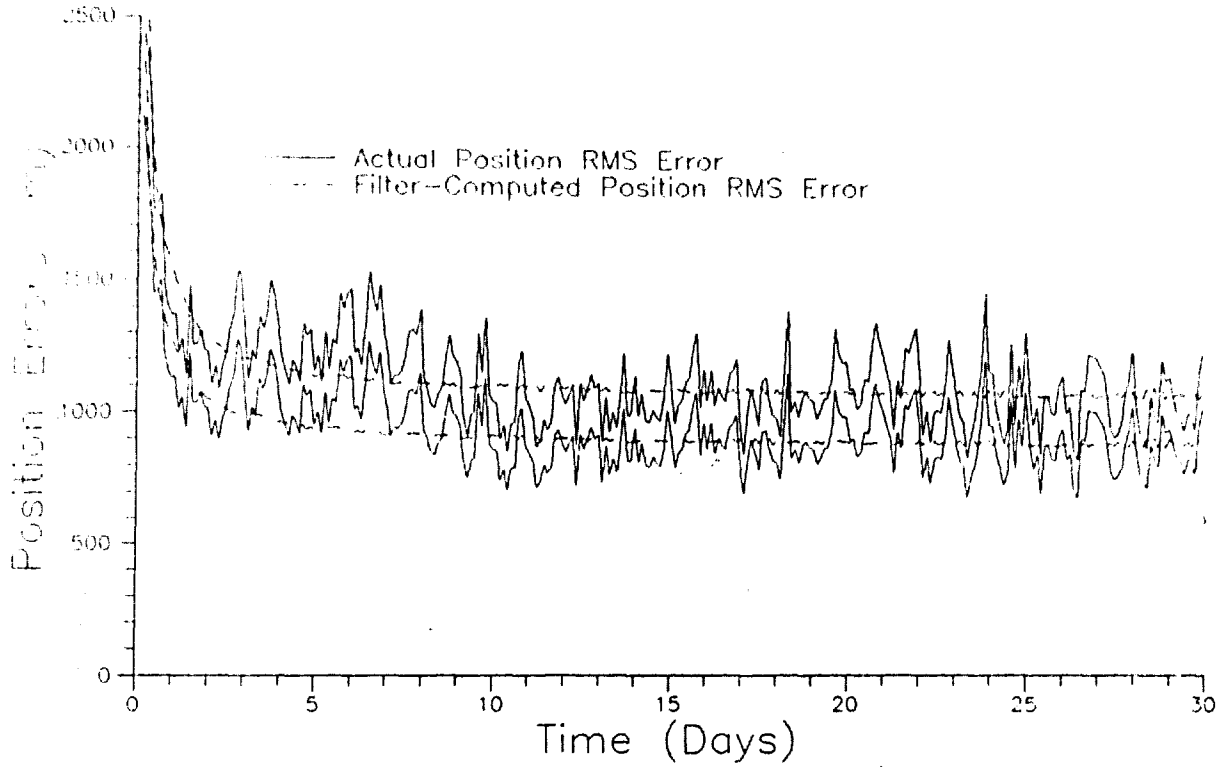

Figure 5-19. Zero-Inclination Orbit Results After Retuning the Filter

Summary of System Characteristics

A generic test orbit was investigated in this chapter to study the behavior of the navigation system proposed. Individual parameters were changed and the effect on system performance examined. The following is a list of the major findings:

(1) The best practical configuration incorporates an observation *at least* every 1.68 minutes, employs instruments with a composite precision of 0.01°, and sights a star near 22° out of the orbit plane. Such a configuration is capable of precisions comparable to the space sextant while making use of only currently available, off-the-shelf hardware.

(2) The performance changes with eccentricity variations, improving as $e \to 0$. The optimum target angle is essentially unchanged for all eccentricities and retuning with changing eccentricities is not necessary.

(3) Changes in inclination can affect the convergence and care must be taken to retune the filter if the inclination changes dramatically. With proper tuning, convergence is possible at all inclinations. Position error increases as $i \Rightarrow 0$, reaching 1 km (after retuning) for the baseline configuration.

The next chapter applies the filter to three practical orbits: low-Earth, semisynchronous, and synchronous. Lessons learned from investigations in this chapter can be directly applied in these situations.

VI. Results of the Short-Term Estimation Problem

The studies in the previous chapter investigated only one contrived "generic" orbit. While quite a few valuable results were produced, the question of how this system performs when applied to more "useful" orbits remains. This chapter addresses the question through Monte Carlo studies of the system when applied to three particularly useful orbits: low-Earth, semisynchronous, and synchronous.

Applied System Configuration

Excellent results were found in the last chapter when a "best practical configuration" was simulated by employing:

(1) An observation rate of 0.01 Hz

(2) $\sigma_{comp} = 0.01^{\circ}$

(3) An empirically determined optimum target angle

Monte Carlo studies require the entire estimated trajectory to be stored for a large number of runs. Since the orbits in this chapter were studied for long periods of time, a 0.01 Hz data rate proved impractical. This is *only* due to on-line storage limitations of the Elxsi minisupercomputer employed in the simulations. *When implemented onboard a spacecraft, very little storage is required.* It was found that output was manageable when the data rate was set to an observation every 3.36 min (~0.005 Hz) and ten Monte Carlo simulations performed. Increasing the number of simulations beyond ten produced little change

in the observed statistics. The use of a 0.005 Hz data rate produces *conservative* estimates of the system's capabilities and higher rates could produce better results.

Low-Earth Orbit Results

In low-Earth orbit, satellites experience significant drag effects. The proposed U.S. Space Station Freedom is an important example of a low-altitude, high drag situation (15), so it was selected for study. Table 6-1 lists the station parameters used in this research.

Table 6-1. Space Station Freedom Data

Initial Orbital Elements:	Classical	Equinoctial
	a = 6785.58 km	a_f = 0
	e = 0	a_g = 0
	M = 10° (1)	a = 6785.58 km
	i = 28°	L = 55°
	ω = 0° (1)	$\chi \approx 0.1763$
	Ω = 45° (1)	$\psi \approx 0.1763$
True Initial Position Error:	10.4 km (1σ)	
True Inital Velocity Error:	10.4 km/week (1σ)	
Coeff. of Drag:	2.22	
Ballistic Coeff.:	49.9 kg/m^2	
Effective Mass:	219,600 kg	
Effective Area:	1980 m^2	
Data Rate:	0.005 Hz	
σ_{comp}:	0.01°	
θ_{target}:	20° Out of orbit plane	

(1) Values arbitrary

(Adapted from Ref. 15)

It was found the filter could be tuned with the pseudonoise matrix $CQ_nC^T = \text{diag}\left\{4.0 \times 10^{-12},\ 4.0 \times 10^{-12},\ 4.1 \times 10^{-4}\ \text{km}^2,\right.$ $\left.1.0 \times 10^{-10}\ \text{rad}^2,\ 9.1 \times 10^{-13},\ 2.5 \times 10^{-13}\right\}$ to match the actual (true) RMS error to the filter-computed value. Using results from the last chapter as a starting point in a search, the optimum target star angle was found to be 20° out of the orbit plane. Air drag necessitates a reboost of the station every 90 days, so it was not necessary to investigate a longer period of time (15).

RMS error results are shown in Figure 6-1, and Figure 6-2 breaks the actual RMS error into orbit-averaged components (along $\hat{e}_r$, $\hat{e}_v$, $\hat{e}_h$). Since the orbital period in this orbit is short and includes few observation points (~27), the data was smoothed over five orbital periods instead of one (as in the last chapter) to clarity of the figures for presentation. The mean steady-state actual (true) RMS error is 534 m, while the filter predicts its RMS error to be 554 m. As would be expected when drag is significant, the in-track component of position yields the largest errors.

Another, often unconsidered, aspect of low-Earth orbit is the effect occasional large solar flares can have on the motion of a spacecraft. Particularly large flares can cause the density of the upper atmosphere to rise to several times its normal value for several days (25:19; 46). Figure 6-3 illustrates a good mathematical model for the flare's effect on the density of Earth's upper atmosphere (31:125-129; 25:19; 46).

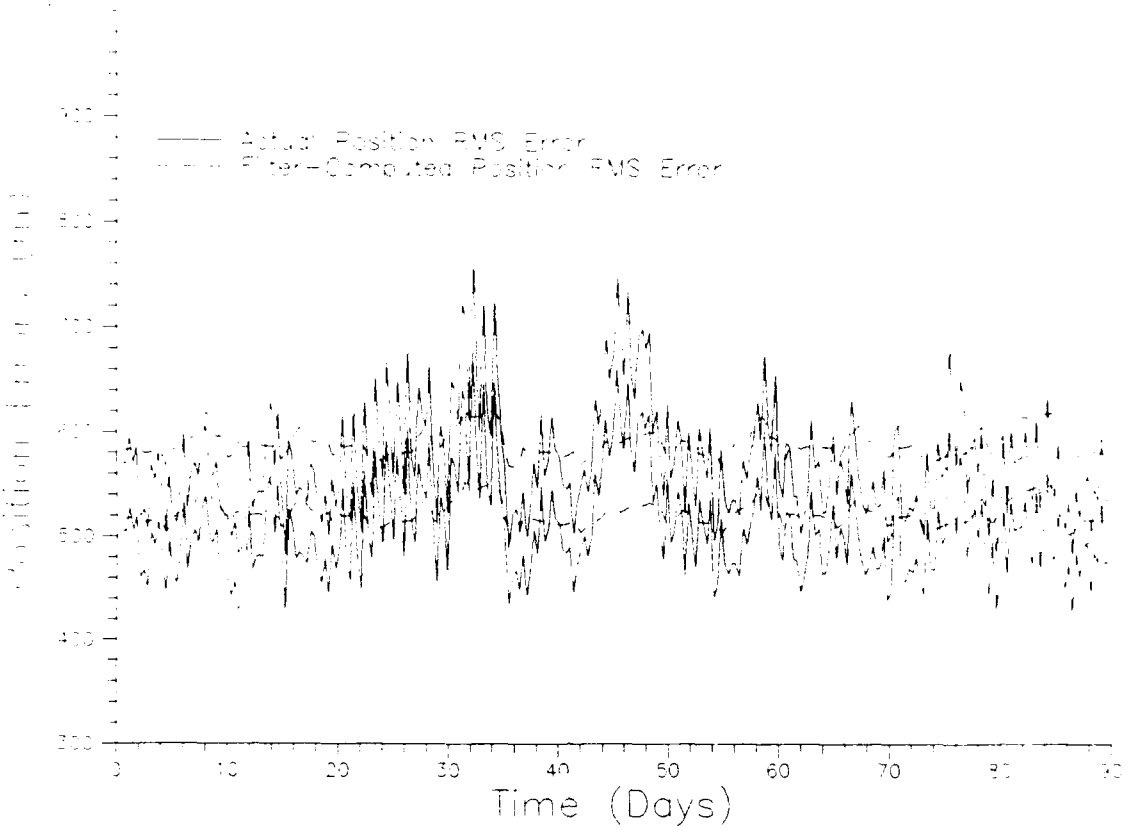

Figure 6-1. Monte Carlo Results for Space Station Freedom

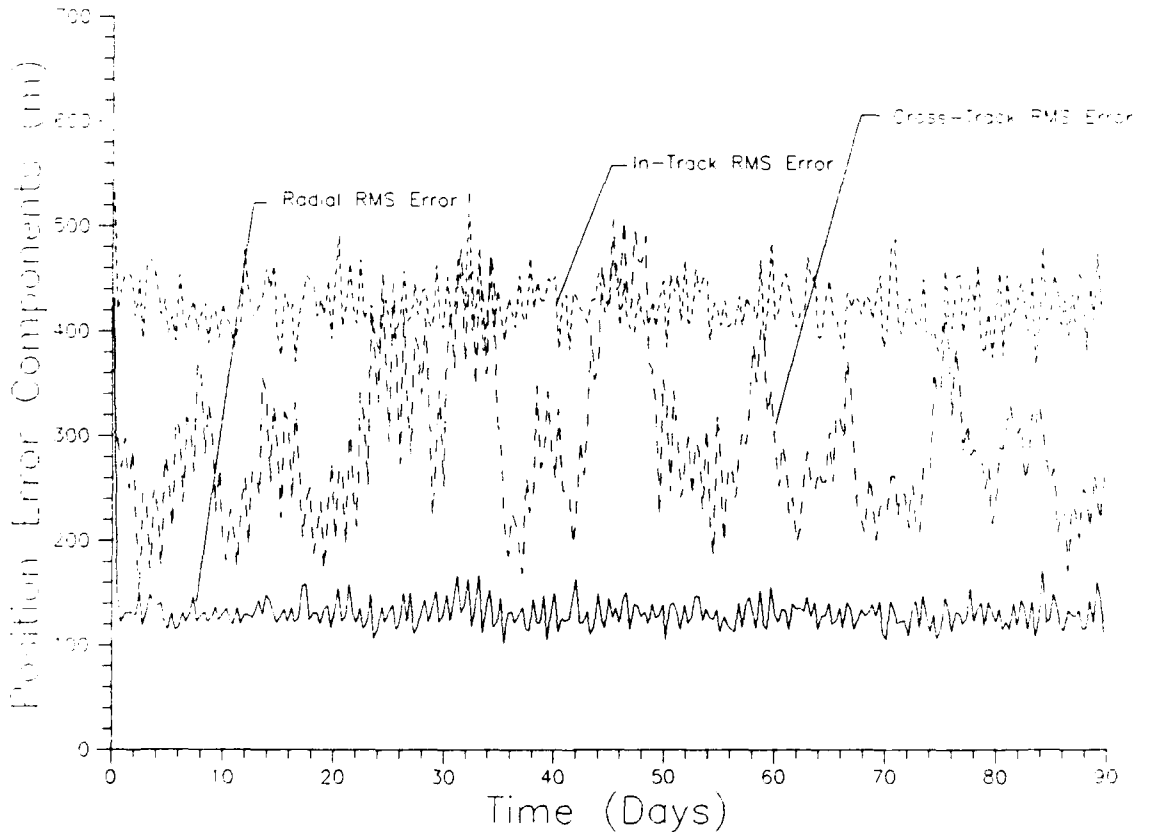

Figure 6-2. Actual (True) RMS Error Components For Freedom

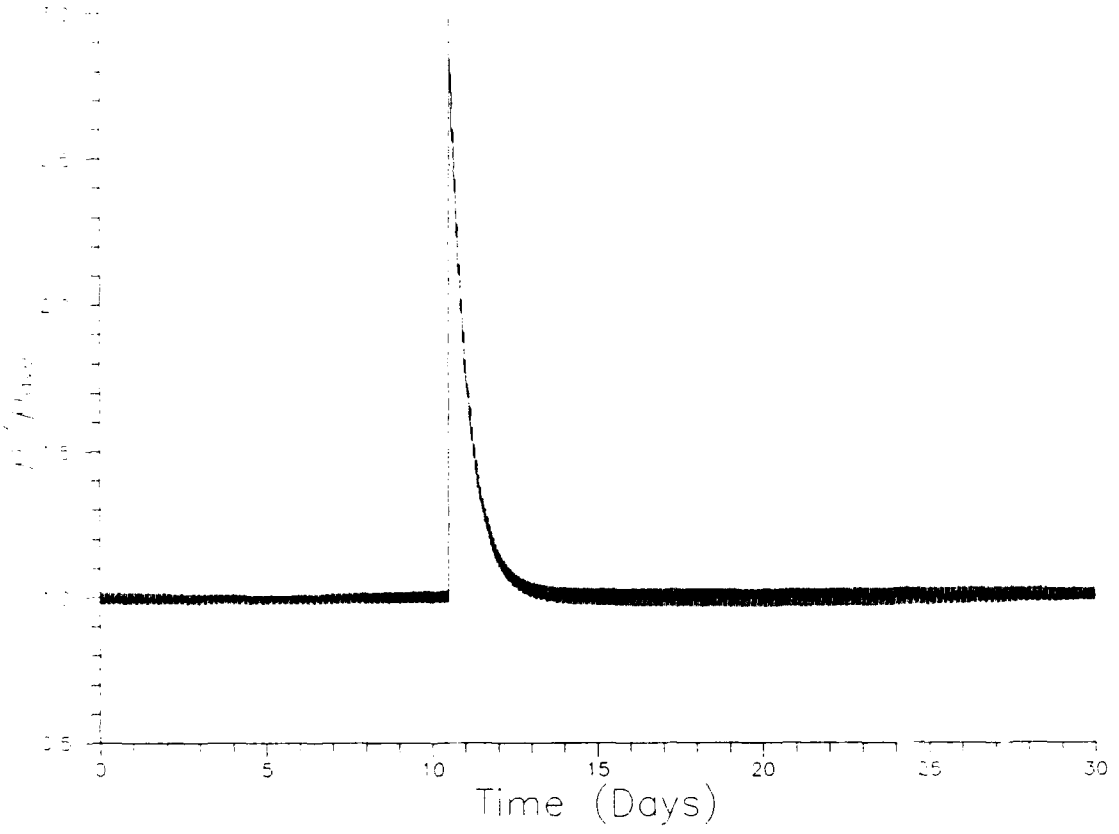

Figure 6-3. Mathematical Approximation to a Solar Flare's Effect on Density in Earth's Upper Atmosphere

A simulation of this navigation system operating during such flare activity was investigated. Figure 6-4 presents the Monte Carlo results with the density profile "seen" by the satellite plotted below for reference. Notice, the mean steady-state RMS error shows no statistically significant difference from the previous situation during the time of interest. Thus, the position estimation is essentially oblivious to even large "one-shot" perturbations.

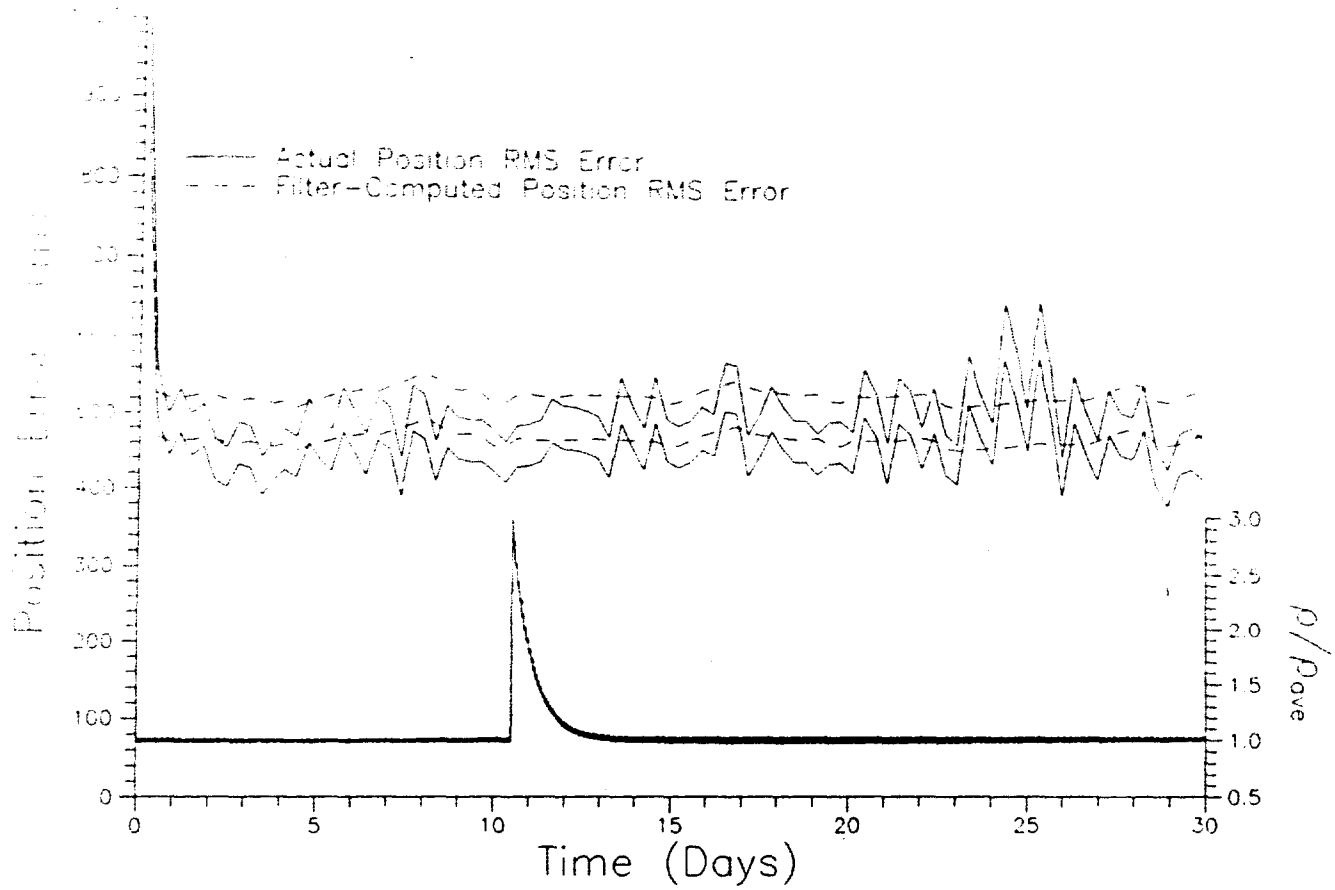

Figure 6-4. Freedom's Position RMS Errors Across a Solar Flare's Effects

Semisynchronous Orbit Results

The Global Positioning System (GPS) is required for many proposed navigation schemes, each relying on the semisynchronous GPS satellites, which, in turn must know their own ephemerides (14:32,34-35; 22; 30:3-4). Ground-based stations presently gather navigation data from each GPS satellite, process it, and transmit updates back to the spacecraft. In the event of ground station failures, the GPS system would begin to fail within a week as the stored ephemerides onboard each satellite begin to depart from reality (14:34; 17:4). As a backup for such a failure, each GPS satellite could employ an autonomous navigation system of its own.

In turn, a user of the GPS system might suffer some loss in precision, but he would not lose use of the system altogether.

The GPS satellite simulated is detailed in Table 6-2. The U-D filter was initially tuned with the pseudonoise matrix $CQ_nC^T = \text{diag}\left\{8.0 \times 10^{-13},\ 8.0 \times 10^{-16},\ 3.7 \times 10^{-3}\ \text{km}^2,\right.$ $\left. 4.0 \times 10^{-14}\ \text{rad}^2,\ 1.0 \times 10^{-11},\ 1.0 \times 10^{-11}\right\}$ to produce conservative behavior. Figure 6-5 presents what appears to be unacceptable performance. Recall, however, that the semisynchronous GPS orbits are resonant with Earth's sectorial harmonics of order 2 and 4 (17:80). These, as

Table 6-2. Semisynchronous Satellite (GPS/NAVSTAR) Data

Initial Orbital Elements:	Classical	Equinoctial
	$a = 26{,}560.24$ km	$a_f = 0$
	$e = 0$	$a_g = 0$
	$M = 0^\circ$	$a = 26{,}560.24$ km
	$i = 55^\circ$	$L = 0^\circ$
	$\omega = 0^\circ$	$\chi = 0$
	$\Omega = 0^\circ$	$\psi \approx 0.5206$
True Initial Position Error:	11.8 m (1σ)	
True Initial Velocity Error:	3.0 m/day (1σ)	
Coeff of Drag:	2.0	
Effective Mass:	2000 kg	
Effective Area:	10 m^2	
Data Rate:	0.005 Hz	
σ_{comp}:	0.01°	
θ_{target}:	12° Out of orbit plane	

(Orbital data adapted from 17:69,80)

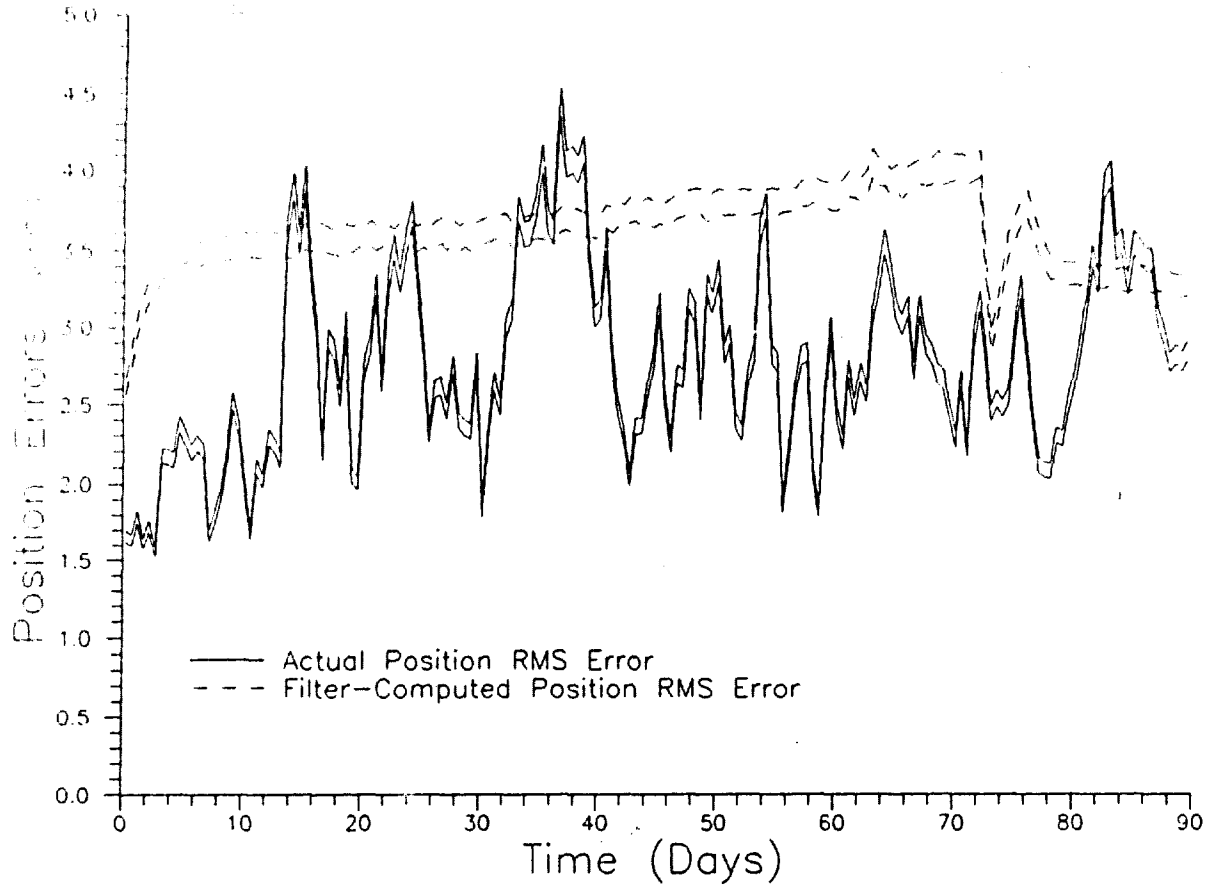

Figure 6-5. Monte Carlo Results for Semisynchronous Orbit

well as the third-body perturbations of the Sun and Moon, are not modelled in Aksnes' theory and can be expected to affect the error in position estimates significantly. Indeed, a plot of the power spectral density for the variance in the radial component of error confirms the contributions due to these unmodelled gravity perturbations.

Figure 6-6 reveals periodic contributions to the error occurring an integer number of times per orbit. Since the orbital period is exactly half the Earth's period of rotation, this trend can be associated in part with the unmodelled sectorial harmonic terms of the Earth's gravity field. At intervals of approximately one orbital period, unmodelled

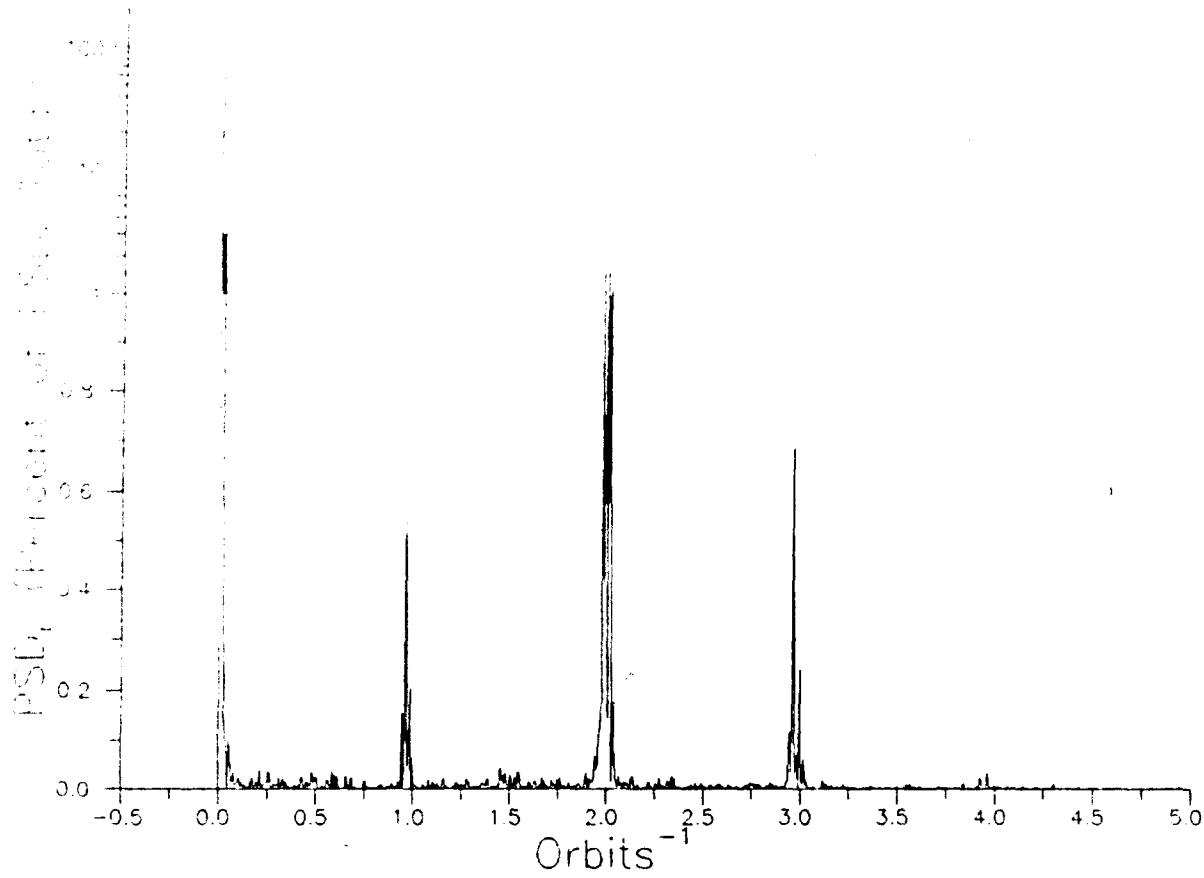

Figure 6-6. Power Spectral Density for Radial Variance

perturbations from the Sun (and, separately, the Moon) can be observed in the plot. Finally, the figure reveals that the majority (72.5%) of the "power" in this error component is due to random bias (frequency = 0 orbits^{-1}) (32:143, 183-184).

Removing the unmodelled effects from the *truth model* allows the effects of *adding* them to the *onboard dynamics* to be *approximated*. Since the onboard dynamics more closely approximate the "real world," the pseudonoise matrix can be reduced to $CQ_nC^T = \text{diag}\left\{8.0 \times 10^{-14}, 8.0 \times 10^{-17}, 3.7 \times 10^{-6}\ \text{km}^2, 4.0 \times 10^{-15}\ \text{rad}^2, 1.0 \times 10^{-14}, 1.0 \times 10^{-14}\right\}$ and still produce quite conservative results as shown in Figure 6-7.

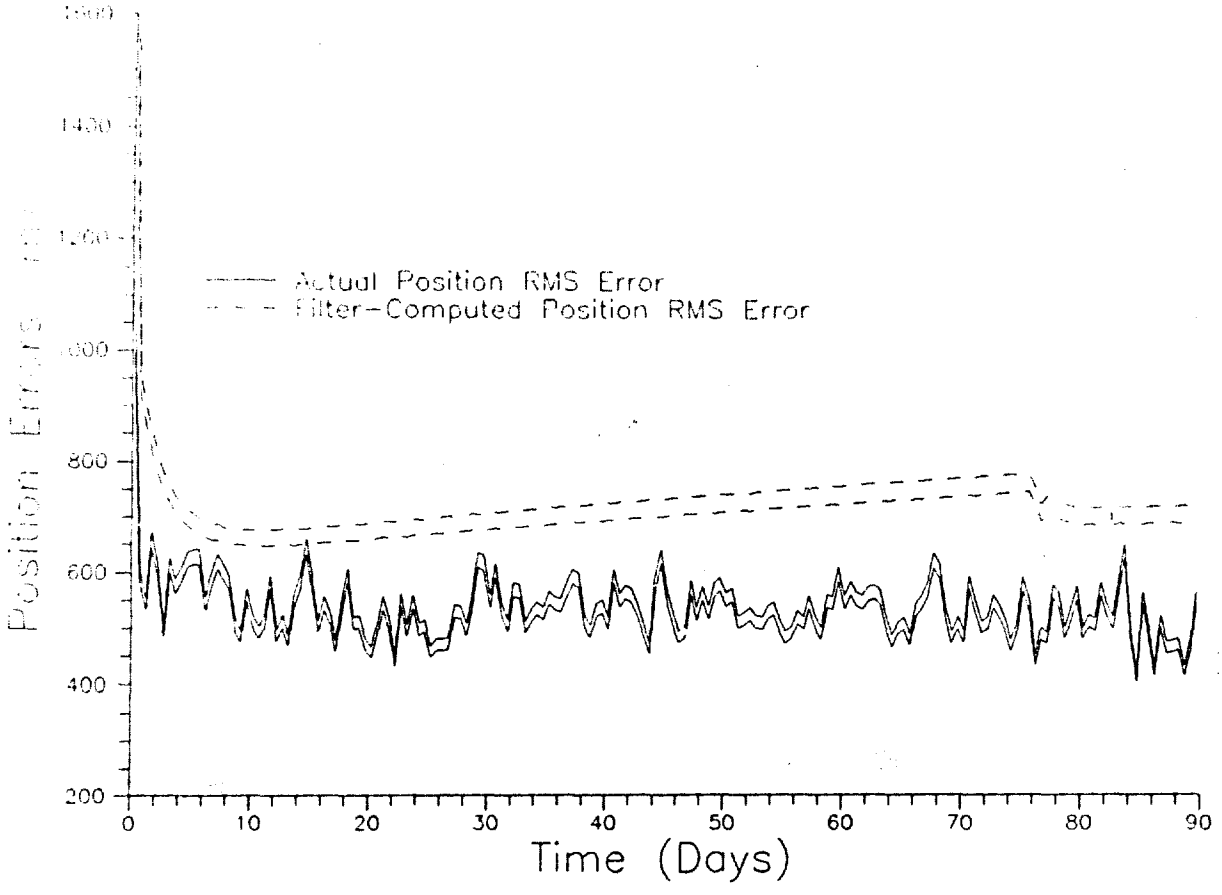

Figure 6-7. Monte Carlo Results for Semisynchronous Orbit (Matched Dynamics)

For reference, a power spectral density plot analogous to Figure 6-6 is presented for this "matched dynamics" case in Figure 6-8.

These figures demonstrate that an onboard model incorporating gravity terms dominant in this particular orbit could produce position estimates precise to at least 500 m. The majority of the power in the error is still random bias (69.8%), with the only other significant contribution occurring twice per orbit (14.2%). The latter source is associated with the modelling of the Earth's oblateness (J_2) in the

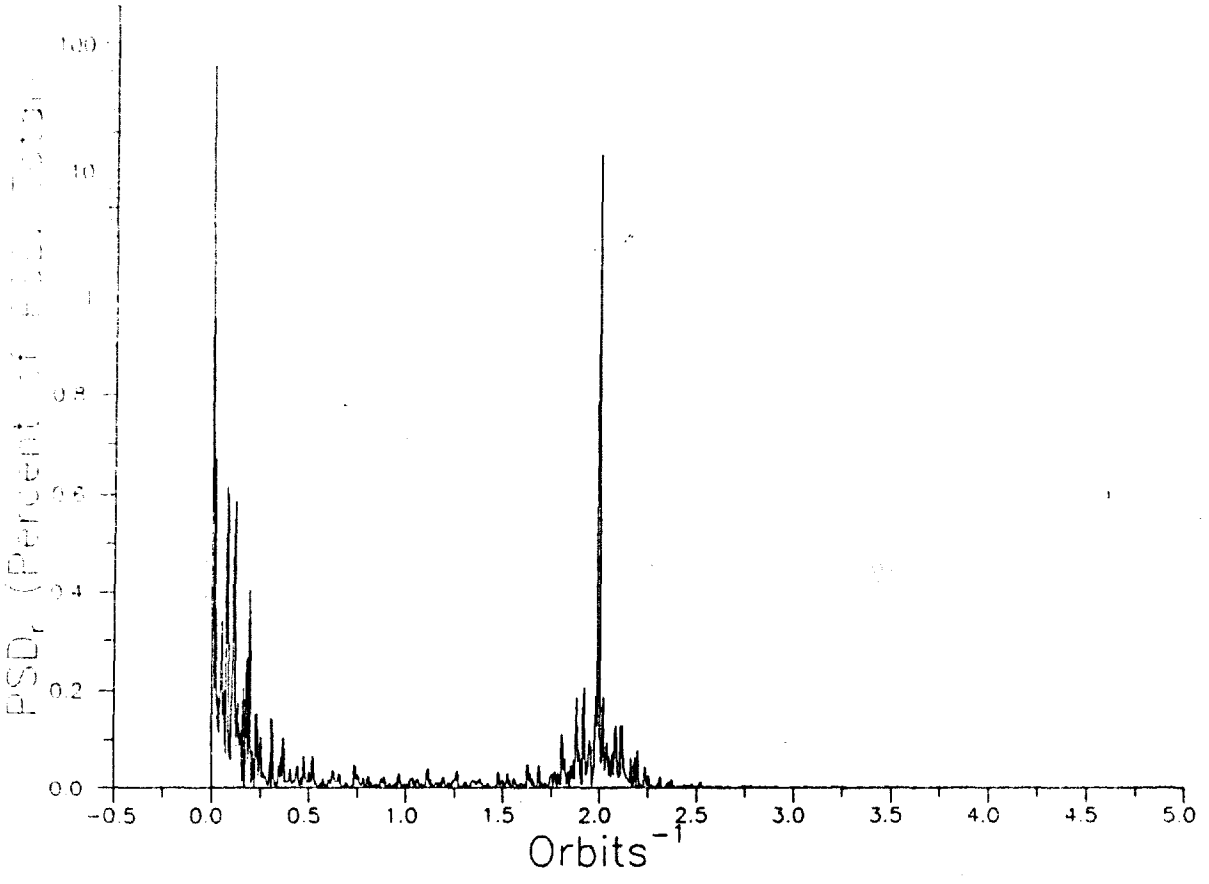

Figure 6-8. Power Spectral Density for Radial Variance (Matched Dynamics)

onboard dynamics and could possibly be reduced with a better dynamics model. Regardless, the precision obtained is sufficient to indicate the usefulness of a system such as this to provide backup navigation information to the GPS satellites.

Synchronous Orbit Results

Commercial satellites in synchronous orbits could greatly benefit from an autonomous navigation system. The reductions in ground support possible directly translate to cost savings for the user. For this reason, the satellite in Table 6-3 was simulated and the Monte Carlo

Table 6-3. Synchronous Satellite Data		
Initial Orbital Elements:	Classical	Equinoctial
	$a = 42,241.09$ km	$a_f = 0$
	$e = 0$	$a_g = 0$
	$M = 10^\circ$	$a = 42,241.09$ km
	$i = 0^\circ$	$L = 100^\circ$
	$\omega = 40^\circ$	$\chi = 0$
	$\Omega = 50^\circ$	$\psi = 0$
True Initial Position Error:	10.4 km (1σ)	
True Initial Velocity Error:	10.4 km/week (1σ)	
Coeff. of Drag:	2.0	
Effective Mass:	2000 kg	
Effective Area:	10 m^2	
Data Rate:	0.005 Hz	
σ_{comp}:	0.01°	
θ_{target}:	17.5° Out of orbit plane	

results are presented in Figure 6-9. The filter was tuned with $CQ_nC^T = \text{diag}\left\{1.0 \times 10^{-14},\ 3.0 \times 10^{-13},\ 3.7 \times 10^{-1}\ \text{km}^2,\right.$ $\left. 5.0 \times 10^{-11}\ \text{rad}^2,\ 4.0 \times 10^{-11},\ 4.0 \times 10^{-12}\right\}$ in this situation.

The large error spikes are due to poor star selection. Recall, the last chapter made mention of a possible ambiguity in radial position when the Earth, spacecraft, and target star are aligned ($\gamma = 0$ or $\gamma = \pi$) for extended periods. For synchronous orbits, the satellite may "linger" in this arrangement for over an hour, allowing the error to grow. Add to this possible simultaneous unmodelled perturbations of the

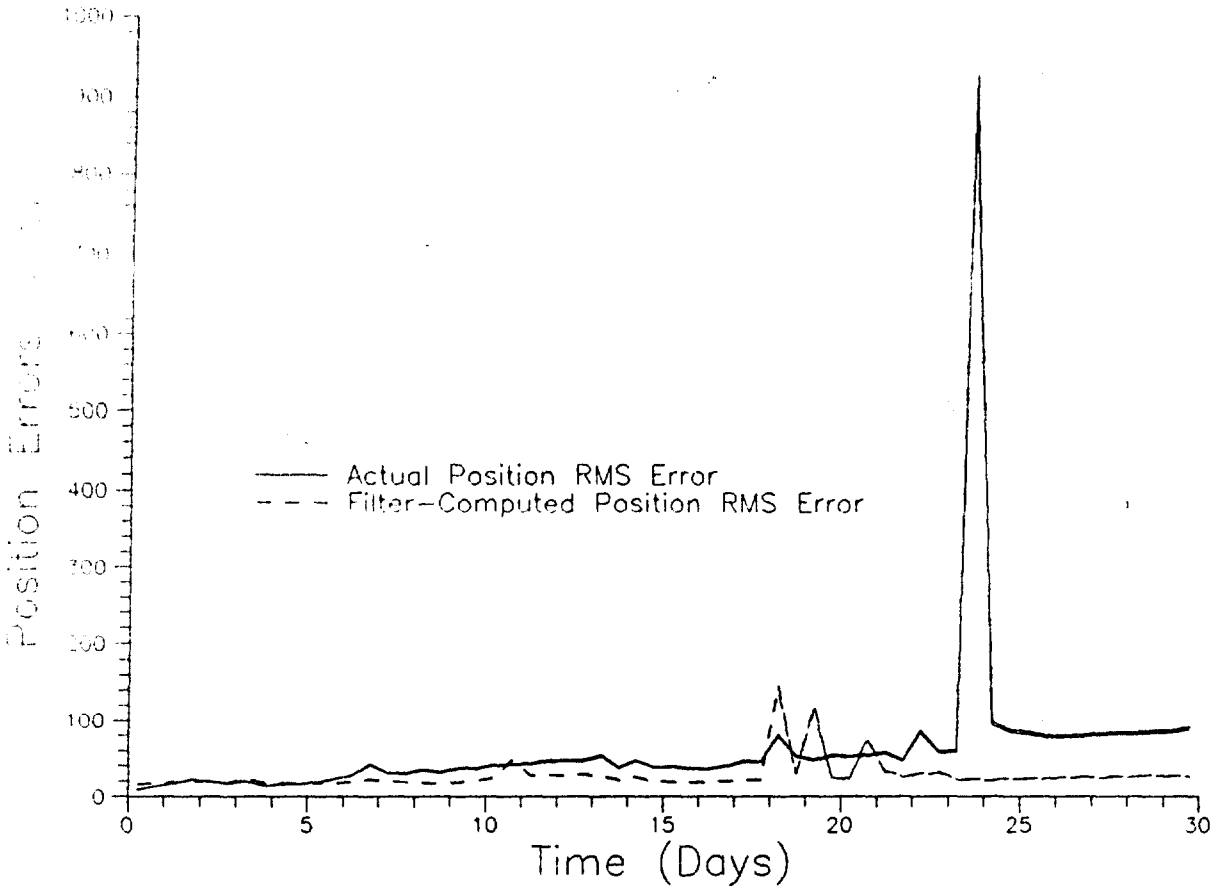

Figure 6-9. Monte Carlo Results for Synchronous Orbit With Poor Star Selection

Sun and Moon, and the position estimates can deteriorate rapidly. The spike at $t \approx 24$ days demonstrates how quickly the error in the estimate can grow in this situation.

Such error growth can be circumvented easily, however. By only allowing the observation of stars at least 25° off the spacecraft-Earth line ($|\cos\gamma| \leq 0.9$), this problem is avoided as shown in Figure 6-10. The 120 days simulated represents the approximate time required for such a spacecraft to drift 1.5° in longitude (-1100 km), creating the necessity for an orbital correction (1:82).

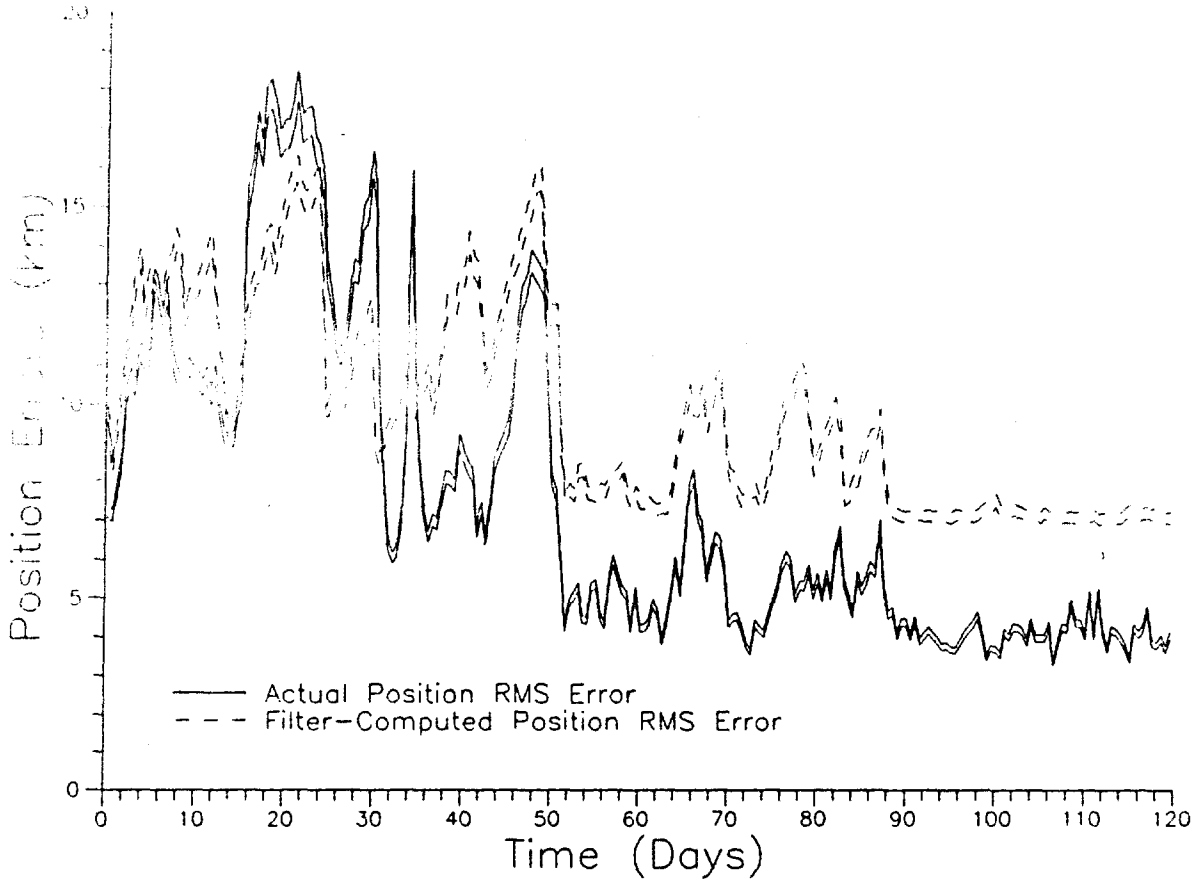

Figure 6-10. Monte Carlo Results for Synchronous Orbit With Proper Choice of a Single Observed Star

The remaining large deviations in the plot are due primarily to the unmodelled third-body effects. (a second source for these deviations will be discussed shortly.) Since general perturbations solutions for Sun and Moon effects are readily available, it was appropriate to simulate including these in the onboard dynamics by, once again, removing them from "reality." Figure 6-11 shows that 5 km precision is easily obtained once these terms are included, even without retuning the filter. These results can be improved still farther.

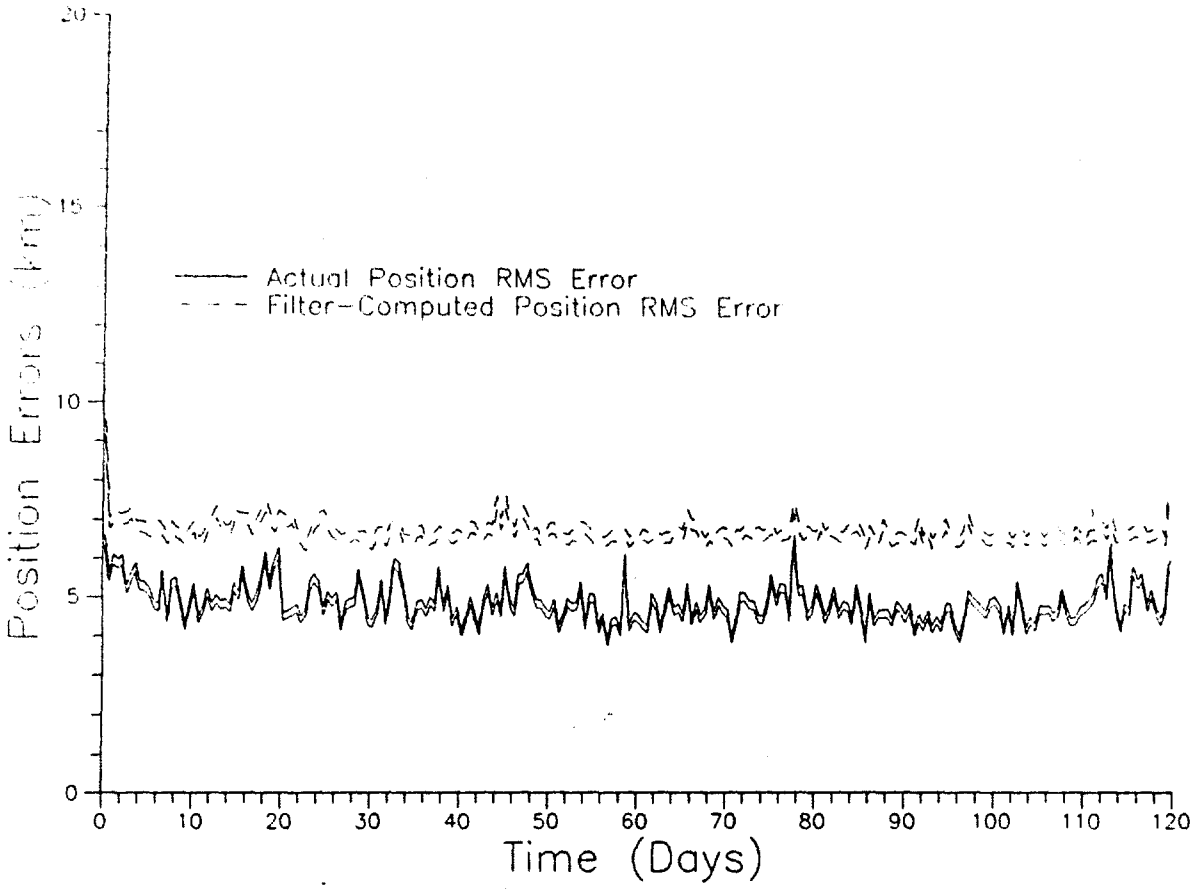

Figure 6-11. Monte Carlo Results for Synchronous Orbit With Proper Choice of a Single Observed Star (No 3rd-Body Effects)

An ambiguity in location, exacerbated by slowly changing measurements, exists for all values of γ. A single angular measurement only establishes the vehicle's position on the surface of an imaginary cone whose apex is at the Earth (8:624-625). Advance knowledge of the spacecraft's approximate position and additional measurements resolve the ambiguity involved. However, if the a priori position estimate is in error and the incoming angular measurements almost identical, the calculated position can become virtually anywhere on the fictitious cone. Therefore, a growth in the estimated position error can be expected when

the angular measurements are changing slowly (as they do in synchronous orbits).

If, instead of always observing the visible star nearest $\theta_{target} = 17.5^{\circ}$, the nearest *two* are alternately sighted, the position can be estimated with more certainty. This added certainty allows the filter to be retuned with $CQ_nC^T = \text{diag}\left\{1.0 \times 10^{-14},\ 3.0 \times 10^{-13},\right.$ $\left. 3.7 \times 10^{-3}\ km^2,\ 5.0 \times 10^{-11}\ rad^2,\ 4.0 \times 10^{-11},\ 4.0 \times 10^{-12}\right\}$. Retuning the filter is, in effect, equivalent to changing the process noise in the dynamics. Retuning the dynamics just because the measurements change seems inconsistent at first glance, so this action deserves a few words of justification.

The matrix CQ_nC^T has, to this point, been viewed as a representation of uncertainty in the dynamics between two consecutive updates. Perhaps a better interpretation is that the matrix represents uncertainty between two updates *that can detect (or observe) an error in the states*. If the error is unobservable, then the update algorithm does not correct the state appropriately when iterating. Two consecutive observations cannot, when $\gamma \approx$ constant, detect a position error until it becomes quite large. Thus, the time between observations which can observe position errors adequately are far apart, as reflected by a large pseudonoise matrix. Viewing two stars alternately increases the observability of state errors; hence, the pseudonoise matrix can be smaller.

Results are shown in Figure 6-12 for the situation in which third-body effects are *not* modelled in the onboard dynamics. For completeness, Figure 6-13 illustrates the performance when the inclusion of

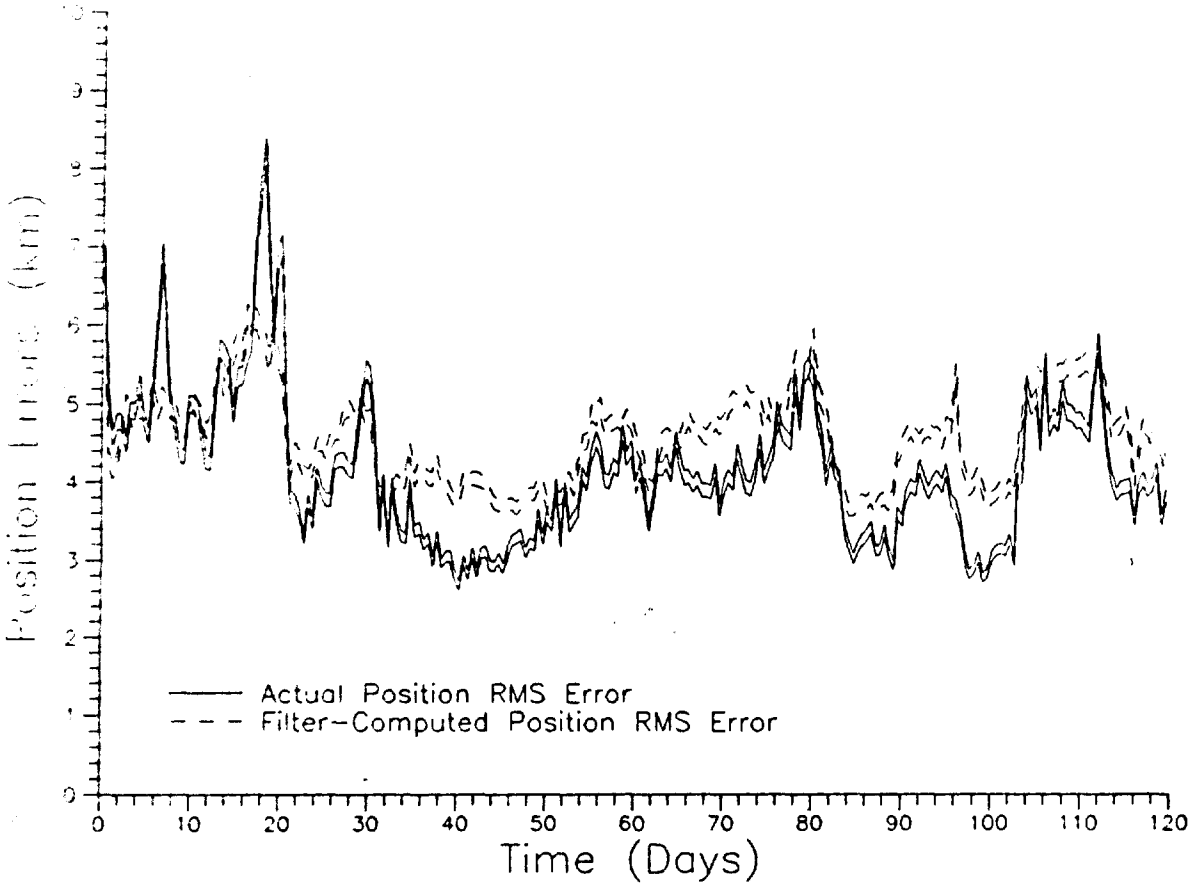

Figure 6-12. Monte Carlo Results for Synchronous Orbit With Two "Best" Stars Observed

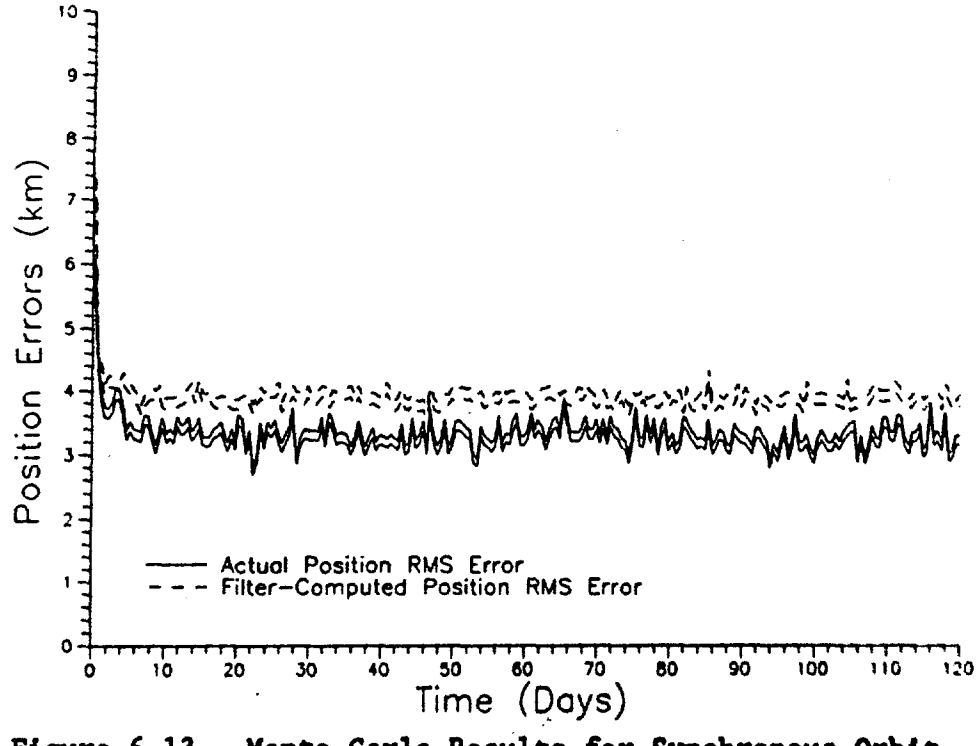

Figure 6-13. Monte Carlo Results for Synchronous Orbit With Two "Best" Stars Observed (No 3rd-Body Effects)

third-body terms in the onboard dynamics is simulated. The observation of two stars again improves the precision of the estimates over one star as can be seen by comparing Figures 6-11 and 6-13.

Alternating between the two "best" stars allows the position of a synchronous satellite to be autonomously estimated to 5 km or less. When one considers synchronous satellites can be allowed to drift hundreds of kilometers between orbit-correction maneuvers, a 5 km error is very acceptable. Therefore, this navigation scheme could be directly applied to such spacecraft with little or no modification.

Conclusion

Results from this section show that the short-term estimation problem (Where Am I?) can be adequately answered with the assumed system configuration. Simple modifications allow the scheme to produce improved performance when necessary. If unmodelled perturbations are significant enough to degrade the performance to an unacceptable level, then Aksnes' model can be improved upon or replaced. Finally, in high-altitude orbits, it "pays" (in improved estimates) to observe more than one star.

VII. The Formulation of the Long-Term Estimation Problem

The ability to predict an orbiting body's future position is a desirable goal for any orbit determination scheme, and autonomous navigation systems are no exception (40:III-66). In the short-term estimation problem, a satellite's position was propagated forward (predicted) over the short periods of time between observations. If not for the effects of drag, that same process could be used to predict far into the future -- the long-term estimation problem. Unmodelled drag effects cause errors in position to grow rapidly when propagating low-Earth trajectories forward over relatively long times. This section derives a process for modelling such effects.

Modifications to Aksnes' Theory to Include Drag Effects

The most observable effect of air drag is an acceleration in the mean anomaly that is quadratic in time; therefore, it is appropriate to modify the dynamics model to account for this effect (23:306; 44:78,95; 45). To a lesser degree, a corresponding linear decrease in the semi-major axis can be observed and must also be included in the theory. An analytical basis for these modifications can be derived.

Ignore, for the moment, the oblateness effects of the primary on the orbit. Further, recall that, for simple two-body motion, the mean anomaly is related to the mean motion by

$$\dot{M} = n = \text{constant} \tag{7-1}$$

where the mean motion is defined as:

$$n = \sqrt{\frac{\mu}{a^3}} \tag{7-2}$$

μ is the gravitational parameter of the primary and a denotes the semimajor axis of the orbit. Further, the mean motion is related to the orbital period by:

$$\mathbb{P} = \frac{2\pi}{n} \tag{7-3}$$

While $\mathbb{P}$ is constant in the two-body problem without drag, it decreases in an approximately linear manner (at least over times which are short compared to the spacecraft's total lifetime) when drag is significant (25:61-62, 71-73; 31:115-116, 118-119; 44:78, 95-97). Thus, let the period be given by:

$$\mathbb{P} = \mathbb{P}_o + \dot{\mathbb{P}}_o \Delta t \tag{7-4}$$

$\mathbb{P}_o$ is the nominal two-body period at epoch $(t = t_o)$ and $\Delta t = t - t_o$ represents the time since epoch. $\dot{\mathbb{P}}_o$ is the rate of change of the period at epoch due to the effects of drag ($\dot{\mathbb{P}}_o < 0$). Incorporating Eq. (7-3) and rearranging, this becomes:

$$n = 2\pi\left(\mathbb{P}_o + \dot{\mathbb{P}}_o \Delta t\right)^{-1} \tag{7-5}$$

Over small times, $\left|\frac{\dot{P}_o \Delta t}{P_o}\right| \ll 1$. Performing a binomial expansion:

$$n \approx \frac{2\pi}{P_o}\left(1 - \frac{\dot{P}_o \Delta t}{P_o}\right) + order\left[\left(\frac{\dot{P}_o \Delta t}{P_o}\right)^2\right] \tag{7-6}$$

From Eq. (7-6), it is apparent that, while n is constant in the two-body problem without drag, it is more appropriate to consider $\dot{n}$ rather than n to be constant when drag exists. Thus, as a first-order approximation, let:

$$\frac{d\dot{M}}{dt} = \dot{n} = \text{constant} \tag{7-7}$$

Since $\dot{P}_o < 0$, Eqs. (7-6) and (7-7) indicate that drag causes an increase (acceleration) in the mean anomaly as compared to the nominal two-body motion. The above equation can be rewritten and integrated:

$$\int d\dot{M} = \int_{t_o}^{t} \dot{n}\, d\tau = \dot{n} \int_{t_o}^{t} d\tau \tag{7-8}$$

Which reduces to:

$$\dot{M} = \dot{n}\left(t - t_o\right) + c \tag{7-8}$$

For this equation to reduce to Eq. (7-1) in the special case of constant mean motion (no drag), c must be equal to n. Substituting for c in Eq. (7-9):

$$\dot{M} = \dot{n}\left(t - t_o\right) + n \tag{7-10}$$

Rearranging and integrating once more:

$$\int_{M_o}^{M} dM = \int_{t_o}^{t} \dot{n}\left(\tau - t_o\right) d\tau + \int_{t_o}^{t} n \, d\tau \tag{7-11}$$

After factoring the constant $\dot{n}$ out of the first integral and performing a change of variables, Eq. (7-11) becomes

$$\int_{M_o}^{M} dM = \dot{n} \int_{0}^{t-t_o} \xi \, d\xi + \int_{0}^{t-t_o} n \, d\xi \tag{7-12}$$

where $\xi = \tau - t_o$. If the "mean" mean motion is defined by

$$\bar{n} = \frac{1}{t - t_o} \int_{0}^{t-t_o} n \, d\xi \tag{7-13}$$

then, Eq. (7-12) simplifies to

$$M(\Delta t) = M_o + \bar{n} \, \Delta t + \frac{\dot{n}}{2} \Delta t^2 \tag{7-14}$$

where M_o is the mean anomaly at an epoch time of $t = t_o$ and $\Delta t = t - t_o$ has been reintroduced for notational convenience. Note that Eq. (7-14) still satisfies Eq. (7-1) when n is constant.

For simple two body motion, the mean anomaly is given by the relation:

$$M(\Delta t) = M_o + n\,\Delta t$$

$$= M_o + \bar{n}\,\Delta t \qquad (7\text{-}15)$$

Comparing this to Eq. (7-14), the quadratic term is seen to be an estimate for the effect of drag.

Recall, however, all oblateness effects of the primary were ignored in the derivation of Eq. (7-14). In Aksnes' theory, terms due to these effects can be lumped together such that

$$M(\Delta t) = M_o + \bar{n}\,\Delta t + g_M \bar{n}\,\Delta t \qquad (7\text{-}16)$$

where g_M is an orbit-dependent constant (4:71). As an approximation for the effects of drag, this equation is modified to reflect the quadratic term in Eq. (7-14):

$$M(\Delta t) = M_o + \bar{n}\,\Delta t + g_M \bar{n}\,\Delta t + \frac{\dot{\bar{n}}}{2}\,\Delta t^2 \qquad (7\text{-}17)$$

This is equivalent to adding the term $\left(\frac{\dot{\bar{n}}}{2}\right)t^2$ to the right-hand-side of Eq. (2) in Reference 4 such that

$$M = M_o + \bar{n}t - \frac{3}{4}\gamma\eta\bar{n}\left\{1 - 3c^2 - \frac{1}{32}\gamma\left[10\left(1 - 6c^2 + 13c^4\right)\right.\right.$$

$$\left. - 5\left(5 - 18c^2 + 5c^4\right)e^2 + 16\eta\left(1 - 6c^2 + 9c^4\right)\right.$$

$$\left.\left. - 15\,\gamma_4\left(3 - 30c^2 + 35c^4\right)\right]\right\} t + \left(\frac{\dot{\bar{n}}}{2}\right)t^2 \qquad (7\text{-}18)$$

where

$$c = \cos i \qquad (7\text{-}19a)$$

$$\eta = \sqrt{1 - e^2} \qquad (7\text{-}19b)$$

$$\gamma = J_2\left(\frac{r}{a\left(1 - e^2\right)}\right)^2 \qquad (7\text{-}19c)$$

$$\gamma_4 = \frac{J_4}{J_2^2} \qquad (7\text{-}19d)$$

and a, e, and i are the semimajor axis, eccentricity, and inclination of the orbit, respectively. J_2 and J_4 are zonal harmonics of the primary. A notationally useful way to rewrite Eq. (7-17) [or Eq. (7-18)] is

$$M(\Delta t) = M_{AK}(\Delta t) + \frac{\dot{n}}{2}\Delta t^2 \qquad (7\text{-}20)$$

where:

$$M_{AK}(\Delta t) = M_o + \bar{n}\,(1 + g_M)\,\Delta t \qquad (7\text{-}21)$$

The "AK" subscript denotes a value of the mean anomaly as predicted by Aksnes' theory. Numerical considerations cited in the last chapter dictated the use of equinoctial elements; therefore, this equation should be rewritten in terms of these elements.

Since drag effects on ω and Ω are limited to small periodic oscillations (25:8,9-10), it is sufficient to assume Aksnes' equations for these terms are exact:

$$\omega(\Delta t) = \omega_{AK}(\Delta t) = \omega_o + g_\omega \bar{n}\Delta t \qquad (7\text{-}22a)$$

$$\Omega(\Delta t) = \Omega_{AK}(\Delta t) = \Omega_o + g_\Omega \bar{n}\Delta t \qquad (7\text{-}22b)$$

g_ω and g_Ω are orbit-dependent constants, similar in form to g_M (4:71). Adding the sum $\pi_{AK} = \omega_{AK} + \Omega_{AK}$ to both sides of Eq. (7-21) and noting the mean longitude is given by $L = \pi + M$, transforms the relation to

$$L(\Delta t) = L_{AK}(\Delta t) + \frac{\dot{n}}{2}\Delta t^2 \qquad (7\text{-}23)$$

where:

$$L_{AK}(\Delta t) = (M_o + \omega_o + \Omega_o) + \bar{n}\,(1 + g_M + g_\omega + g_\Omega)\,\Delta t$$

$$= L_o + \bar{n}\,(1 + g_M + g_\omega + g_\Omega)\,\Delta t \qquad (7\text{-}24)$$

The quadratic acceleration in the mean longitude is intimately linked to a corresponding effect on the semimajor axis. Differentiating the definition of the mean motion as given in Eq. (7-2) produces a relationship between the two orbital elements:

$$\frac{\dot{n}}{2} = -\frac{3}{4}\sqrt{\mu}\; a^{-5/2}\,\dot{a} \qquad (7\text{-}25)$$

Since $\frac{\dot{n}}{2}$ is assumed to be constant, it is appropriate to evaluate this relation at the epoch time, t_o:

$$\frac{\dot{n}}{2} = -\frac{3}{4}\sqrt{\mu}\; a_o^{-5/2}\,\dot{a}_o \qquad (7\text{-}26)$$

In the above relation, it is not necessary to subscript the constant $\frac{\dot{n}}{2}$ to denote evaluation at epoch.

Aksnes' theory treats the semimajor axis as a constant, and a non-zero value of $\dot{a}_o$ in Eq. (7-26) would indicate otherwise; therefore, it is appropriate to modify his theory to reflect this fact. A Taylor series expansion on the semimajor axis about the epoch time yields (to first order):

$$a(\Delta t) \approx a_o + \dot{a}_o \Delta t \qquad (7\text{-}27)$$

Or, incorporating Eq. (7-26) and following the format from before, this is written

$$a(\Delta t) = a_{AK}(\Delta t) - \frac{4}{3\sqrt{\mu}} a_o^{5/2} \frac{\dot{n}}{2} \Delta t \qquad (7\text{-}28)$$

where a_{AK} has been expressed as a function of time to reiterate the fact that this value is predicted, albeit as a constant, by Aksnes' routines. This relation indicates the semimajor axis decreases linearly with time in the presence of drag. King-Hele followed a completely different approach and also concluded the semimajor axis is reduced in this manner (25:40).

The self-consistency of the above derivation to this point can be checked. The assumption of a linearly decreasing period, $P = P_o + \dot{P}_o \Delta t$, was introduced in Eq. (7-4) and eventually lead to a linearly decreasing semimajor axis, $a = a_o + \dot{a}_o \Delta t$, as reflected in Eq. (7-28). To verify the consistency of the derivation, it is only necessary to substitute the resulting linear approximation for a back into the definition of the period:

$$P = \frac{2\pi}{\sqrt{\mu}} a^{3/2} \qquad (7\text{-}29)$$

$$\approx \frac{2\pi}{\sqrt{\mu}} a_o^{3/2} \left[1 + \left(\frac{\dot{a}}{a}\right)_o \Delta t\right]^{3/2}$$

$\left(\frac{\dot{a}}{a}\right)_o = \frac{\dot{a}_o}{a_o}$ has been introduced as notational shorthand. Performing a binomial expansion and retaining the terms to first order, this becomes

$$\mathbb{P} \approx \frac{2\pi}{\sqrt{\mu}} a_o^{3/2}\left[1 + \frac{3}{2}\left(\frac{\dot{a}}{a}\right)_o \Delta t\right] \tag{7-30}$$

$$= \frac{2\pi}{n_o} + \frac{3\pi}{n_o}\left(\frac{\dot{a}}{a}\right)_o \Delta t$$

where it has been recognized that $n_o = \sqrt{\frac{\mu}{a_o^3}}$. Thus, since $\left(\frac{\dot{a}}{a}\right)_o \leq 0$, Eq. (7-30) does indeed verify that the derived linear relationship for the semimajor axis is consistent with a linearly decreasing period. Further, by comparing Eq. (7-30) with Eq. (7-4), the following equivalences result:

$$\mathbb{P}_o = \frac{2\pi}{n_o} \tag{7-31a}$$

$$\mathbb{P} = \frac{3\pi}{n_o}\left(\frac{\dot{a}}{a}\right)_o \tag{7-31b}$$

Together, Eqs. (7-31) can be assembled to form:

$$\left(\frac{\dot{\mathbb{P}}}{\mathbb{P}}\right)_o = \frac{3}{2}\left(\frac{\dot{a}}{a}\right)_o \tag{7-32}$$

This is a well known relationship between the period and semimajor axis, directly derivable from the definition of the period without making use

of binomial expansions (31:117). This consistency with a known relationship provides additional confidence in the derivation thus far. In light of Eqs. (7-30) and (7-32), it is apparent the assumptions and approximations are, at least to first order, self-consistent thus far.

Modifications to Aksnes' Theory to Improve the Epoch Estimates

In a perfect world, the modifications to Aksnes' theory shown in Eqs. (7-23) and (7-28) would produce completely acceptable results when propagating trajectories forward over long periods. This is an estimation problem, however, and the epoch values L_o and a_o will, in general, be in error. If the variation of Eq. (7-23) is taken at a specific time, then:

$$\delta L(\Delta t) = \delta L_o + \delta \bar{n} \, (1 + g_M + g_\omega + g_\Omega) \, \Delta t \qquad (7\text{-}33)$$

The constant $(g_M + g_\omega + g_\Omega)$ is of order $\left(\frac{J_4}{J_2}\right)$ (4:71). In other words, $(g_M + g_\omega + g_\Omega)$ is on the order of 10^{-4} and $(1 + g_M + g_\omega + g_\Omega) \approx 1$ for the purposes of this variation; therefore, the following equation relates errors in the mean longitude at time t to errors at epoch:

$$\delta L(\Delta t) = \delta L_o + \delta \bar{n} \, \Delta t \qquad (7\text{-}34)$$

δL_o is clearly the error in the estimate of L_o, while the term $\delta \bar{n}$ requires a bit of manipulation to interpret its physical basis.

The linear approximation for the mean motion can be introduced into the definition of $\bar{n}$

$$\bar{n} \approx \frac{1}{t-t_o} \int_0^{t-t_o} \left(n_o + \dot{n}\xi \right) \, d\xi \tag{7-35}$$

and a variation taken to yield:

$$\delta\bar{n} \approx \frac{1}{t-t_o} \, \delta \int_0^{t-t_o} \left(n_o + \dot{n}\xi \right) \, d\xi \tag{7-36}$$

This is a fixed time problem, so the variational operator can be moved inside the integral. Further, $\dot{n}$ is being considered to be a fixed constant. Thus, Eq. (7-36) becomes

$$\delta\bar{n} \approx \frac{1}{t-t_o} \int_0^{t-t_o} \delta n_o \, d\xi$$

$$= \frac{\delta n_o}{t-t_o} \int_0^{t-t_o} d\xi \tag{7-37}$$

$$= \delta n_o$$

where it has been noted δn_o is independent of time (indeed, δn_o is a parameter evaluated at a specific time, t_o) and can, therefore, be taken outside the integral for the integration.

Replacing $\delta\bar{n}$ with δn_o in Eq. (7-34) produces:

$$\delta L(\Delta t) = \delta L_o + \delta n_o \Delta t \tag{7-38}$$

By taking a variation of the definition $n = \sqrt{\frac{\mu}{a^3}}$ and evaluating at epoch, the linear term above, δn_o, can be related to an error in a_o:

$$\delta n_o = -\frac{3}{2}\sqrt{\mu}\, a_o^{-5/2}\, \delta a_o \tag{7-39}$$

This represents the well-known linear effect on the mean anomaly (or, equivalently, the mean longitude) produced by an error in a_o.

A similar variational argument applied to the semimajor axis produces:

$$\delta a(\Delta t) = \delta a_o \tag{7-40}$$

Or, in light of Eq. (7-39):

$$\delta a(\Delta t) = -\frac{2}{3\sqrt{\mu}}\, a_o^{5/2}\, \delta n_o \tag{7-41}$$

Putting together all of the corrections produces

$$L_{RW}(\Delta t) = L_{AK}(\Delta t) + \delta L_o + \delta n_o \Delta t + \frac{\dot{n}}{2} \Delta t^2 \qquad (7\text{-}42)$$

$$a_{RW}(\Delta t) = a_{AK}(\Delta t) + \Gamma\, \delta n_o + 2\Gamma \left(\frac{\dot{n}}{2}\right) \Delta t \qquad (7\text{-}43)$$

where $\Gamma = -\frac{2}{3\sqrt{\mu}}\, a_o^{5/2}$ has been used as a shorthand notation. The subscript "RW" stands for "real world" because these relationships model the actual dynamics much better than Aksnes' relations and are meant to be a step closer to representing the "real world." Eqs. (7-42) and (7-43) show that the true trajectory departs from that predicted by Aksnes' theory; therefore, given proper values for the constants, $L_{RW}(\Delta t)$ and $a_{RW}(\Delta t)$ would be better suited for propagating the trajectory far into the future. This simply amounts to replacing $L_{AK}(\Delta t)$ and $a_{AK}(\Delta t)$ with $L_{RW}(\Delta t)$ and $a_{RW}(\Delta t)$, respectively, in Aksnes' equations of motion.

Determination and Application of the Correction Terms

It still remains to be shown how the constants δL_o, δn_o, and $\frac{\dot{n}}{2}$ are determined. Of these three, $\frac{\dot{n}}{2}$ is of primary concern since this term embodies all of the principal effects of drag. As will be shown shortly, the epoch correction terms (δL_o and δn_o) are easily obtained in conjunction with the determination of $\frac{\dot{n}}{2}$.

A seemingly plausible, yet erroneous, approach is to include $\frac{\dot{n}}{2}$ as an additional state variable and to estimate it along with the orbital elements. This is undesirable from two aspects. First, increasing the number of state variables causes a corresponding increase in the amount

of time required for the filter to perform state updates and propagations. This extra computation time is difficult to justify at every observation time since the drag parameter $\frac{\dot{n}}{2}$ is only needed for an occasional long-term prediction. Second, and more importantly, when the time between observations is short, the effect of drag cannot be separated from random errors in the measurements. In this situation, the calculation of $\frac{\dot{n}}{2}$ is essentially meaningless. Both of these problems can be avoided; however, by using a two-step process.

During the short-term estimation problem ("Where am I?"), estimates for all orbital elements are generated and can be stored at various times. Then, after a sufficiently long time, these estimates can be examined and the effects of drag $\left(\frac{\dot{n}}{2}\right.$ and, equivalently, $\left.\dot{a}_o\right)$ deduced. This is accomplished by rearranging Eqs. (7-42) and (7-43) as

$$\Delta L_i = L_{RW}(\Delta t_i) - L_{AK}(\Delta t_i) = \delta L_o + \delta n_o \Delta t_i + \frac{\dot{n}}{2} \Delta t_i^2 \qquad (7\text{-}44)$$

$$\Delta a_i = a_{RW}(\Delta t_i) - a_{AK}(\Delta t_i) = \Gamma\, \delta n_o + 2\Gamma \left(\frac{\dot{n}}{2}\right) \Delta t_i \qquad (7\text{-}45)$$

where $\Delta t_i = t_i - t_o$ (the time *before* epoch) and Γ is the previously defined quantity. The "RW" states are those found in the short-term problem -- the "real world" values. The "AK" values are those "predicted" by propagating backwards from the epoch time using Aksnes' theory. Figures 7-1 and 7-2 graphically illustrate the quantities ΔL_i and Δa_i with results from a typical Monte Carlo run. The quadratic and linear trends of ΔL_i and Δa_i are quite clearly shown in these

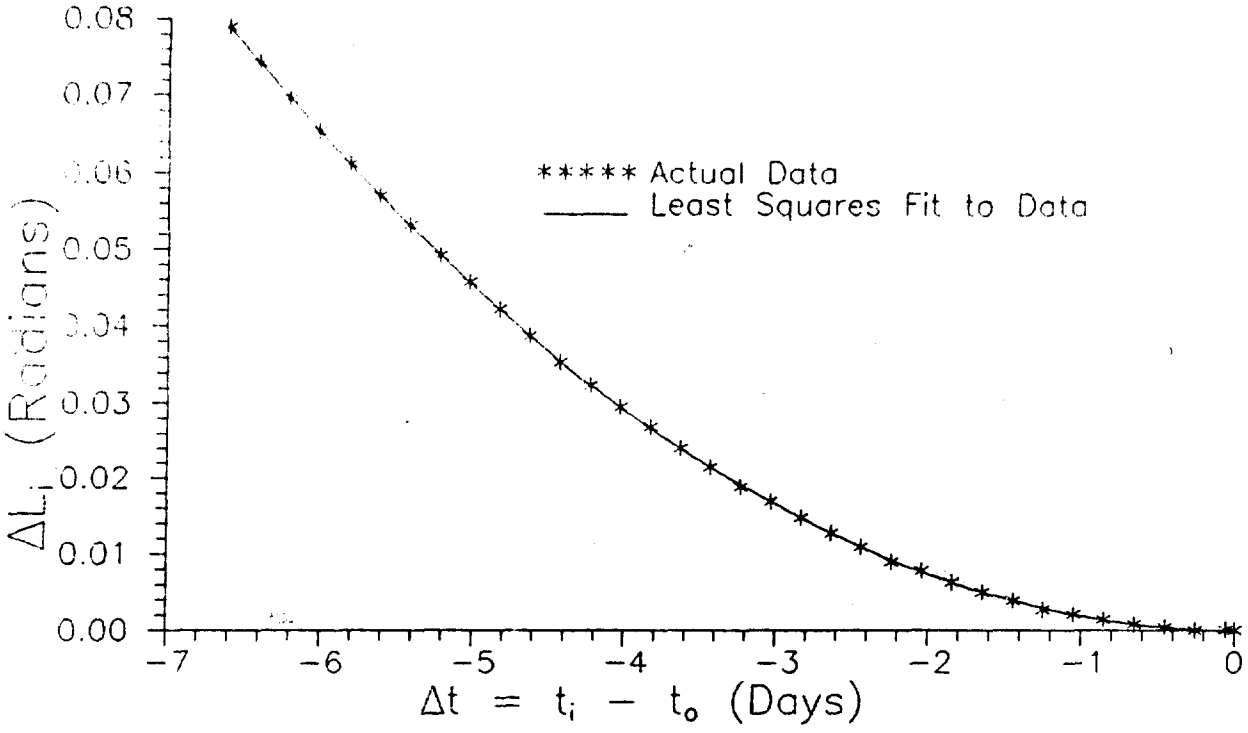

Figure 7-1. Sample ΔL_i Data

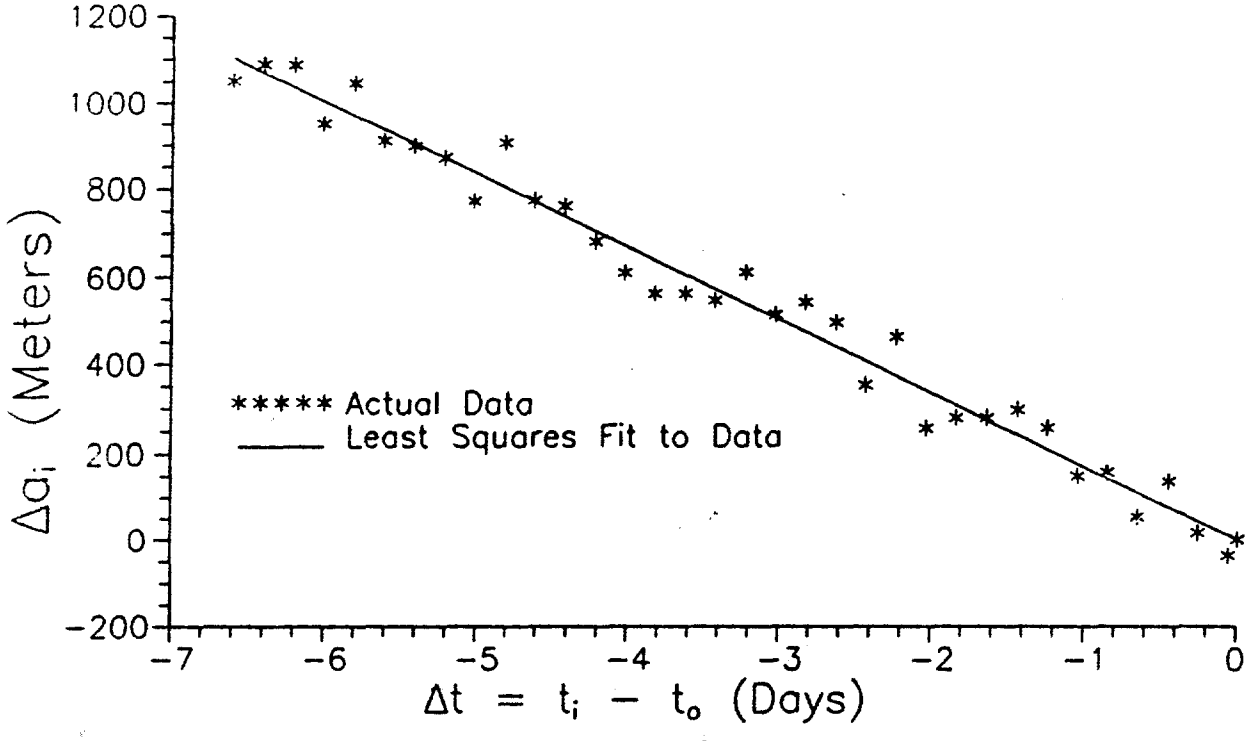

Figure 7-2. Sample Δa_i Data

figures by way of linear least squares (LLS) fits. Indeed, a LLS formulation is an excellent method to determine the constants δL_o, δn_o, and $\frac{\dot{n}}{2}$.

A simple rewriting of Eqs. (7-44) and (7-45) allows the differences between actual and predicted values to be used as data for a LLS routine:

$$\Delta L_i = z_{1i} = \begin{bmatrix} 1 & \Delta t_i & \Delta t_i^2 \end{bmatrix} \begin{bmatrix} \delta L_o \\ \delta n_o \\ \frac{\dot{n}}{2} \end{bmatrix} = T_{1i} X \tag{7-46}$$

$$\Delta a_i = z_{2i} = \begin{bmatrix} 0 & \Gamma & 2\Gamma\Delta t_i \end{bmatrix} \begin{bmatrix} \delta L_o \\ \delta n_o \\ \frac{\dot{n}}{2} \end{bmatrix} = T_{2i} X \tag{7-47}$$

Then, after a sufficient number of data points are collected, the constants of the $\underset{\sim}{X}$ vector are readily computed from:

$$\underset{\sim}{X} = \begin{bmatrix} \delta L_o \\ \delta n_o \\ \frac{\dot{n}}{2} \end{bmatrix} = P_x \sum_{i=1}^{2} \sum_{j=1}^{n} T_{ij}^T Q_{ij}^{-1} z_{ij} \tag{7-48}$$

$P_x = \left[\sum_{i=1}^{2} \sum_{j=1}^{n} T_{ij}^T Q_{ij}^{-1} T_{ij} \right]^{-1}$ is the covariance of the estimated elements of $\underset{\sim}{X}$ and the Q_{ij}'s are the corresponding (scalar) values of the variance

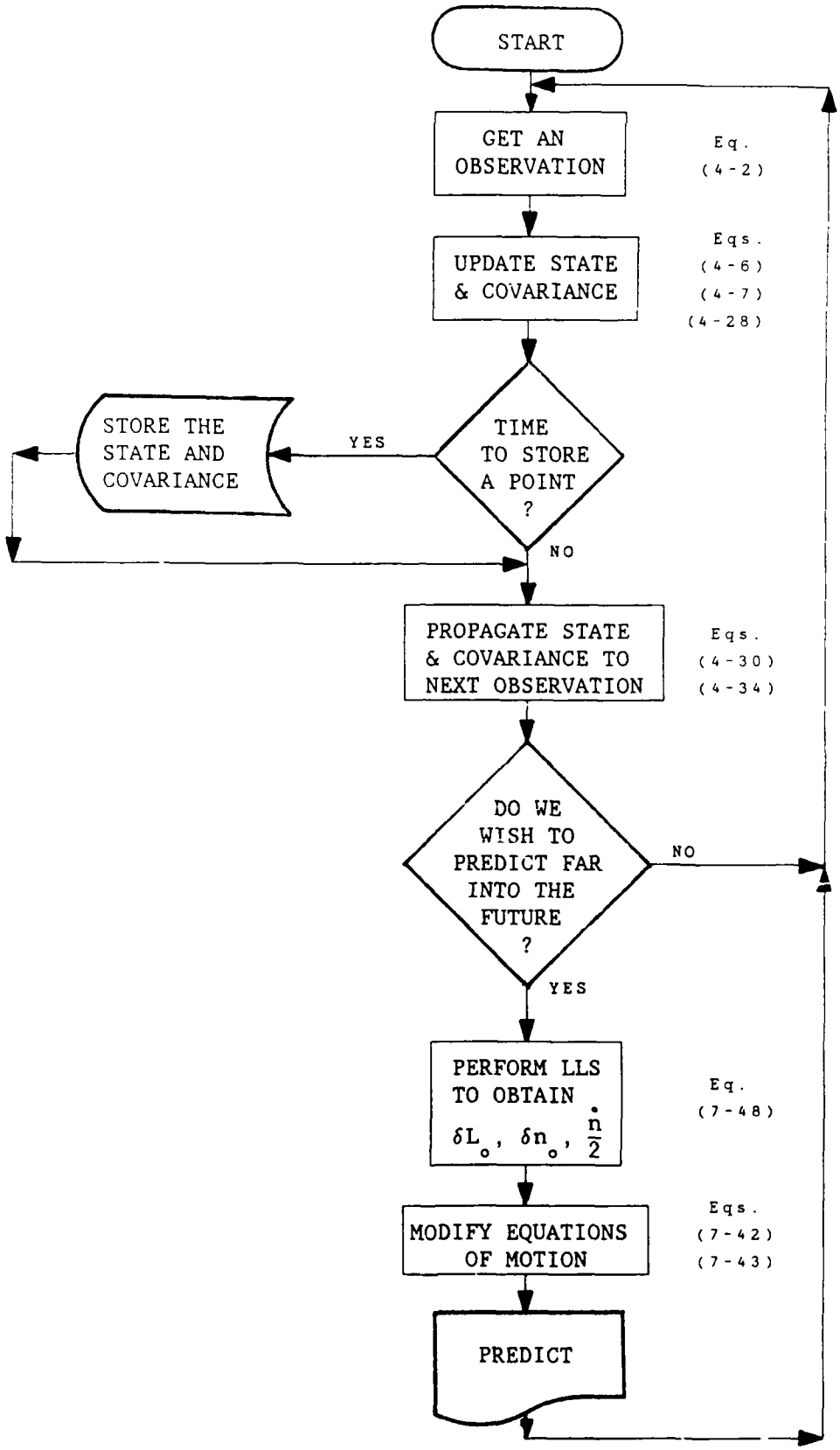

Figure 7-3. Two-Step Algorithm for Position Prediction

for each datum, z_{ij} (36:509-520; 44:56-59). A computer-style algorithm for applying this LLS method is shown in Figure 7-3. Because the drag and epoch correction calculations are removed from the short-term estimation problem, this can be termed a two-step process. The first step is the real-time calculation of the orbital elements and the second a LLS procedure to find the constants δL_o, δn_o, and $\frac{\dot{n}}{2}$ from stored data prior to predicting far into the future.

The term $\Gamma = -\frac{2}{3\sqrt{\mu}} a_o^{5/2}$ involves an imperfect estimate, a_o; in fact, the LLS finds δa_o (through δn_o) in an effort to correct this value. This small error causes very little effect on the value of Γ; never the less, a slight improvement in the calculation of the other parameters in $\underline{X}$ can be realized by incorporating an optional calculation. This step is an initial refinement of a_o prior to performing the LLS described in Eq. (7-48).

In addition to improving the estimate of Γ, refining the initial estimate of a_o has a secondary benefit. If, in Eq. (7-44), the linear term $\delta n_o \Delta t$ is large relative to other terms, the difference could become $\Delta L \approx \delta n_o \Delta t$ due to numerical truncation; i.e., all information about the mean longitude error (δL_o) and drag $\left(\frac{\dot{n}}{2}\right)$ *could be lost*. Since this term is proportional to the error in the semimajor axis at epoch [through Eq. (7-39)], an initial correction would reduce the relative size of it when the main LLS calculation is performed. Thus, a better initial estimate for a_o will also help minimize the chances of losing information about δL_o and $\frac{\dot{n}}{2}$ due to numerical truncation.

A preliminary correction term, δa_o, can be calculated from the data involving only the semimajor axis, Δa_i. Then, the value for a_o (and, therefore, Γ) can be improved before solving for the entire vector of constants, $\underset{\sim}{X}$. This is accomplished by noting that Eq. (7-45) is really just a rewriting of

$$\Delta a_i = a_{RW}(\Delta t_i) - a_{AK}(\Delta t_i) = \delta a_o + a_o \Delta t_i \tag{7-49}$$

so a least squares solution can be written as

$$\underset{\sim}{Y} = \begin{bmatrix} \delta a_o \\ a_o \end{bmatrix} = P_Y \sum_{i=1}^{n} T_i^T Q_{2i}^{-1} \Delta a_i \tag{7-50}$$

where:

$$P_Y = \left[\sum_{i=1}^{n} T_i^T \, Q_{2i}^{-1} \, T_i \right]^{-1} \tag{7-51a}$$

$$T_i = \begin{bmatrix} 1 & \Delta t_i \end{bmatrix} \tag{7-51b}$$

The resulting correction term, δa_o, can then be used to improve the estimate of the semimajor axis at epoch. Evaluating Eq. (7-49) at epoch and rearranging allows this correction to be applied:

$$a_{RW}(0) = a_{AK}(0) + \delta a_o \tag{7-52}$$

Or, letting $a_{AK}(0)$ be the initial estimate of a_o, this can be rewritten

$$a(0^+) = a(0^-) + \delta a_o \tag{7-53}$$

where the minus and plus superscripts represent conditions before and after correction, respectively. The relevant portion of the two-step algorithm, modified to include this step, is shown in Figure 7-4.

A few words of warning about the Q_{ij}'s are needed here. A quite attractive approach to incorporating these would be to combine the appropriate elements from the covariance matrix in the filter and from a covariance matrix propagated back from epoch using Aksnes' theory. Recall, when two scalars x and y, with associated uncertainties σ_x and σ_y are subtracted, the uncertainty of their difference is the simply:

$$\sigma_{x-y} = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{7-54}$$

The differences formed in Eqs. (7-44) and (7-45) are, similarly scalars, and the corresponding uncertainties are the square roots of the appropriate elements from the covariance matrices. Thus:

$$Q_{1i} = \sigma^2_{\Delta L_i} = \left(Q_{L_i L_i}\right)_{RW} + \left(Q_{L_i L_i}\right)_{AK} \tag{7-55}$$

$$Q_{2i} = \sigma^2_{\Delta a_i} = \left(Q_{a_i a_i}\right)_{RW} + \left(Q_{a_i a_i}\right)_{AK} \tag{7-56}$$

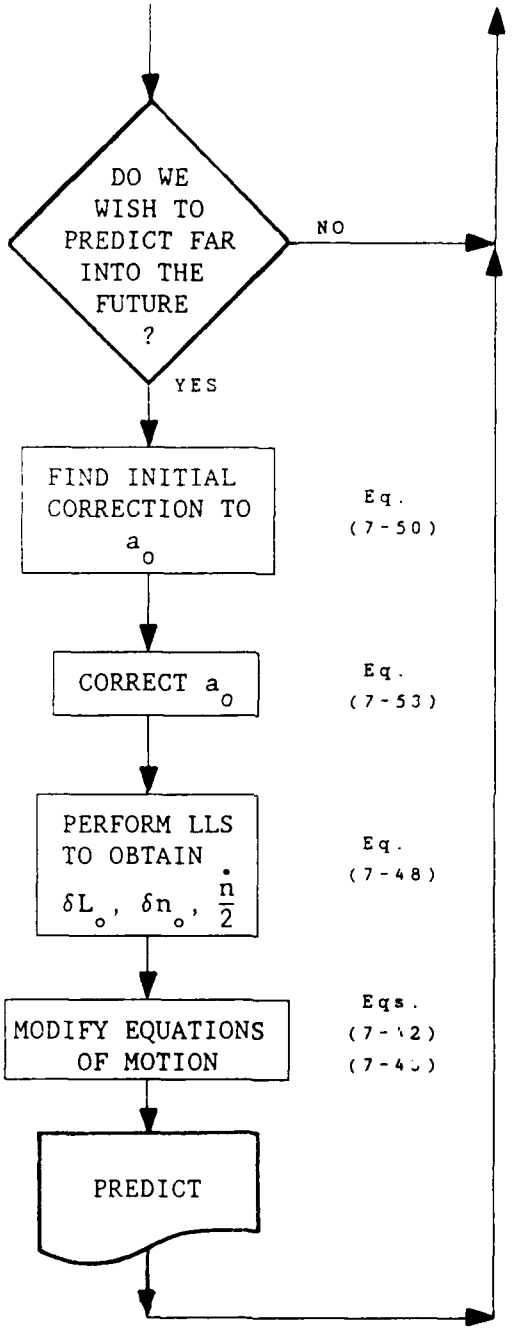

Figure 7-4. Lower Portion of Two-Step Algorithm With Optional Correction Step Shown

This approach should be used with *extreme* caution, however, as it was found this tended to overweight the points closest to the epoch. The resulting value of $\frac{\dot{n}}{2}$ overpredicted the effect of drag. Recall from Chapter IV that the state transition matrix used to propagate the covariance [Eqs. (4-30), (4-36), (4-37)] was found numerically. Errors in such a calculation can become larger as the time of propagation grows, causing $\left(Q_{L_i L_i}\right)_{AK}$ and $\left(Q_{a_i a_i}\right)_{AK}$ to become unrealistically large. This situation was avoided by arbitrarily weighting all data equally $(Q_{ij} = 1)$ with excellent results.

Summary

During the course of the long-term estimation problem, the navigation system produces state estimates at every data point, a portion of which can be stored. Then, at a later time, general trends in stored values of the mean longitude and semimajor axis can be examined. This yields information not only about drag effects, but also about errors in the current (epoch) state. Together, this knowledge allows the state to be predicted forward over long periods of time. Figures 7-3 and 7-4 summarize this two-step process. A detailed application of this method is presented in the next section.

VIII. Results of the Long-Term Estimation Problem

The two-step process derived in the previous section has been applied to the proposed space station Freedom. The same parameters as in the short-term estimation problem were assumed and are repeated in Table 8-1 for reference. In this chapter, results for "routine" predictions as well as some worst-case scenarios involving the effects of solar flares are presented. Where applicable, methods for minimizing the impact of flares on the predictor performance are presented.

Routine Predictions

Routine predictions are those that take place when a "typical" Earth atmosphere is present. In terms of the "real world" as assumed by the truth model, this means that the spacecraft is subjected to an oblate, rotating, exponential atmosphere. In other words, the atmosphere departs little from the average during the time of interest and unusually large disturbances (solar flare effects, etc.) are far apart.

The first scenario explored was one in which Freedom computed its current position for eight days and then predicted its future position from that point forward, without the aid of additional data. Thirty-five points spread over seven days were used in the two-step algorithm to estimate δL_o, δn_o, and $\frac{\dot{n}}{2}$.

Figure 8-1 very graphically illustrates the superiority of the two step process. Applying uncorrected Aksnes' equations directly produced an RMS error of 1100 km at the end of 12 days. This was reduced an order

Table 8-1. Space Station Freedom Data

Initial Orbital Elements:	Classical	Equinoctial
	$a = 6785.58$ km	$a_f = 0$
	$e = 0$	$a_g = 0$
	$M = 10^\circ$ (1)	$a = 6785.58$ km
	$i = 28^\circ$	$L = 55^\circ$
	$\omega = 0^\circ$ (1)	$\chi \approx 0.1763$
	$\Omega = 45^\circ$ (1)	$\psi \approx 0.1763$

True Initial Position Error:	10.4 km (1σ)
True Inital Velocity Error:	10.4 km/week (1σ)
Coeff. of Drag:	2.22
Ballistic Coeff.:	49.9 kg/m^2
Effective Mass:	219,600 kg
Effective Area:	1980 m^2
Data Rate:	0.005 Hz
σ_{comp}:	0.01°
θ_{target}:	20° Out of orbit plane

Tuning Matrix:

$$CQ_nC^T = \text{diag}\left\{4.0 \times 10^{-12},\ 4.0 \times 10^{-12},\right.$$
$$4.1 \times 10^{-4}\ \text{km}^2,\ 1.0 \times 10^{-10}\ \text{rad}^2,$$
$$\left. 9.1 \times 10^{-13},\ 2.5 \times 10^{-13}\right\}.$$

(1) Values arbitrary

(Adapted from Ref. 15)

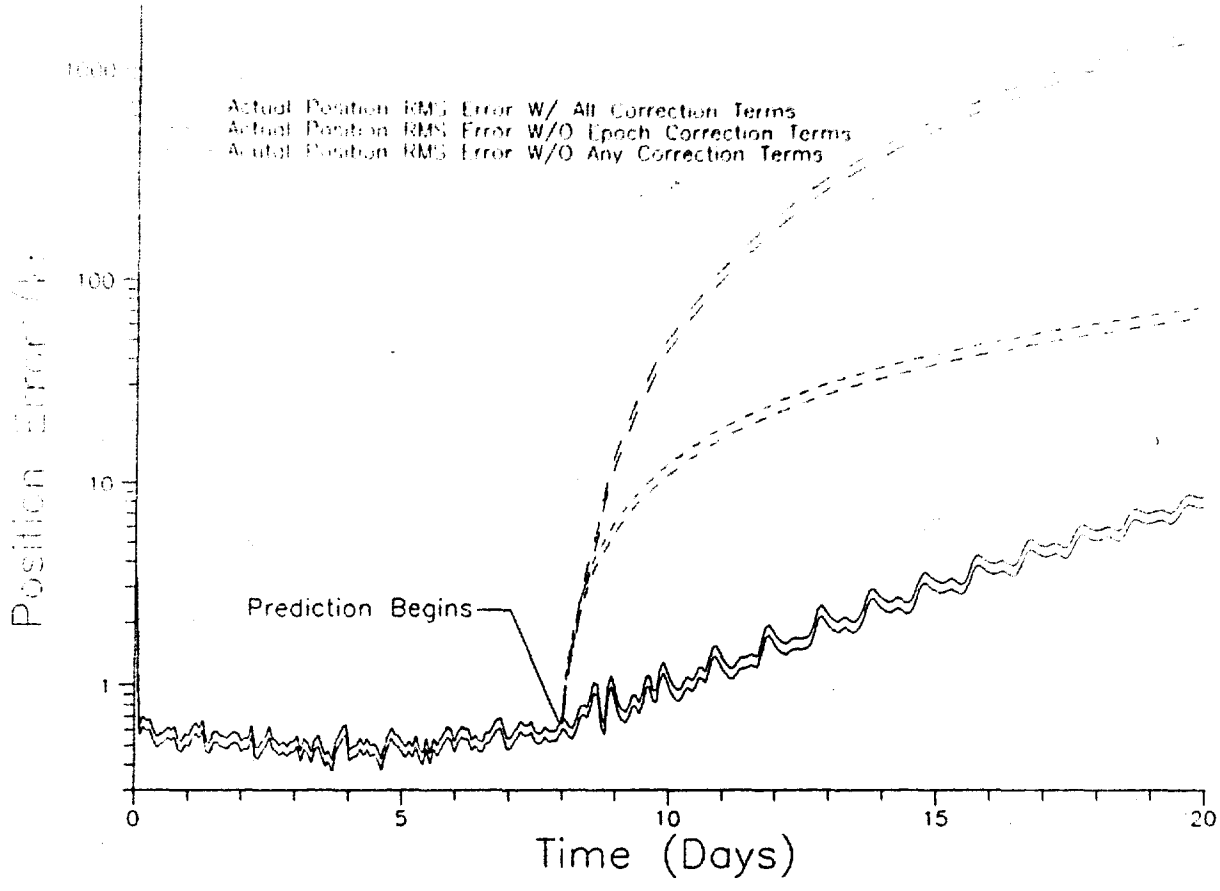

Figure 8-1. Comparison of Prediction Methods

of magnitude by including the drag correction term $\frac{\dot{n}}{2}$ in the equations of motion [Eqs. (7-23) and (7-28)] as the second curve in the figure shows. The best results, however, were produced by including not only $\frac{\dot{n}}{2}$ but also the epoch correction terms δL_o and δn_o [Eqs. (7-42) and (7-43). This method allows better precision for longer prediction times. Ground-based orbit determination facilities routinely produce estimates to around 12 km -- Figure 8-2 shows that this system can easily *predict two weeks* into the future with that precision (42:63).

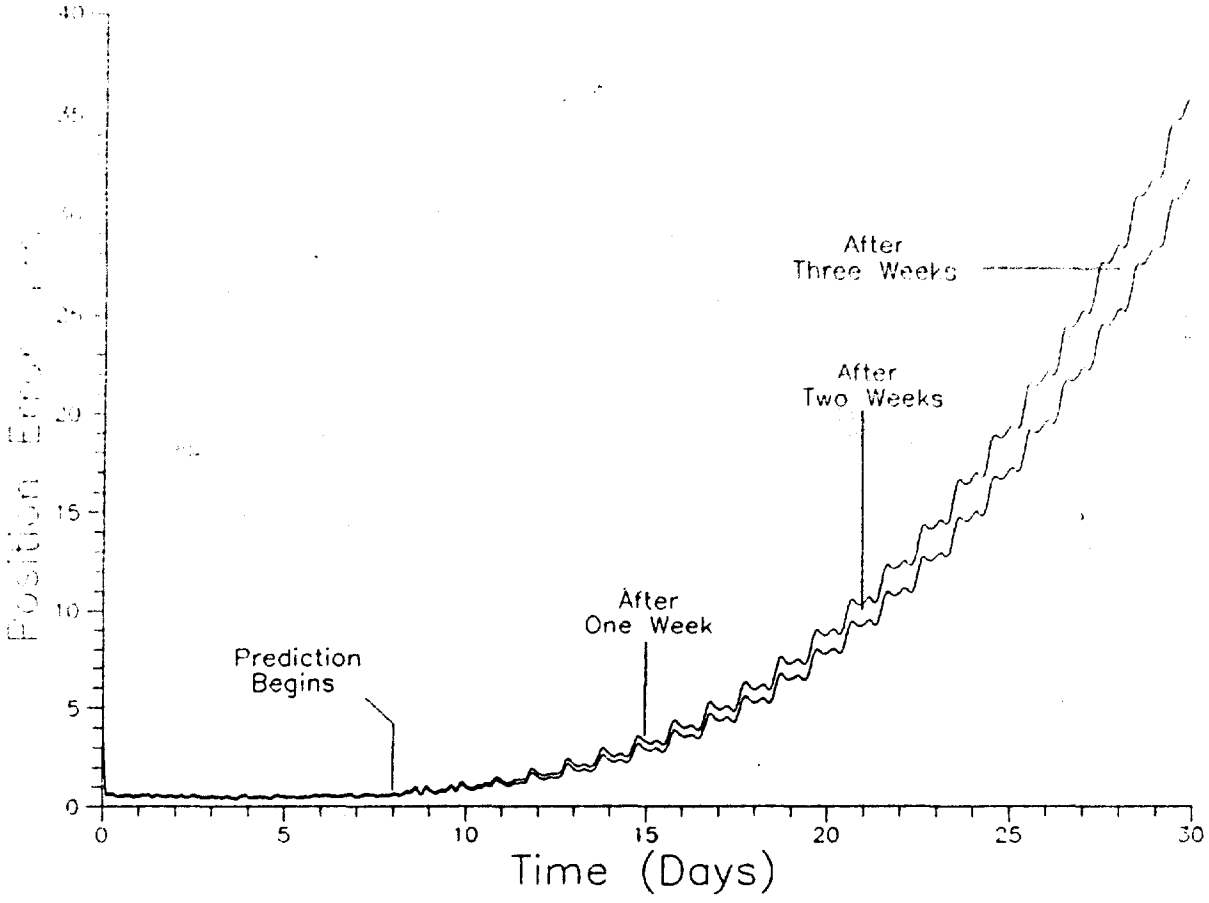

Figure 8.2 Actual (True) RMS Error With Two-Step Method

Figure 8-3 plots the individual components of RMS error, revealing that the quadratic growth in total RMS error seen in the previous figure is the result of quadratic growth in the in-track position error. The in-track component of error grows, in part, due to the inability to calculate $\frac{\dot{n}}{2}$ exactly. The number of points used in the calculation of this term could be increased, but was deemed unnecessary from two aspects.

First, additional points require additional storage and computations. When resources are limited, it is best to limit the number of data points stored and processed to a minimum. Second, and more importantly, there is no real advantage gained in finding a better value --

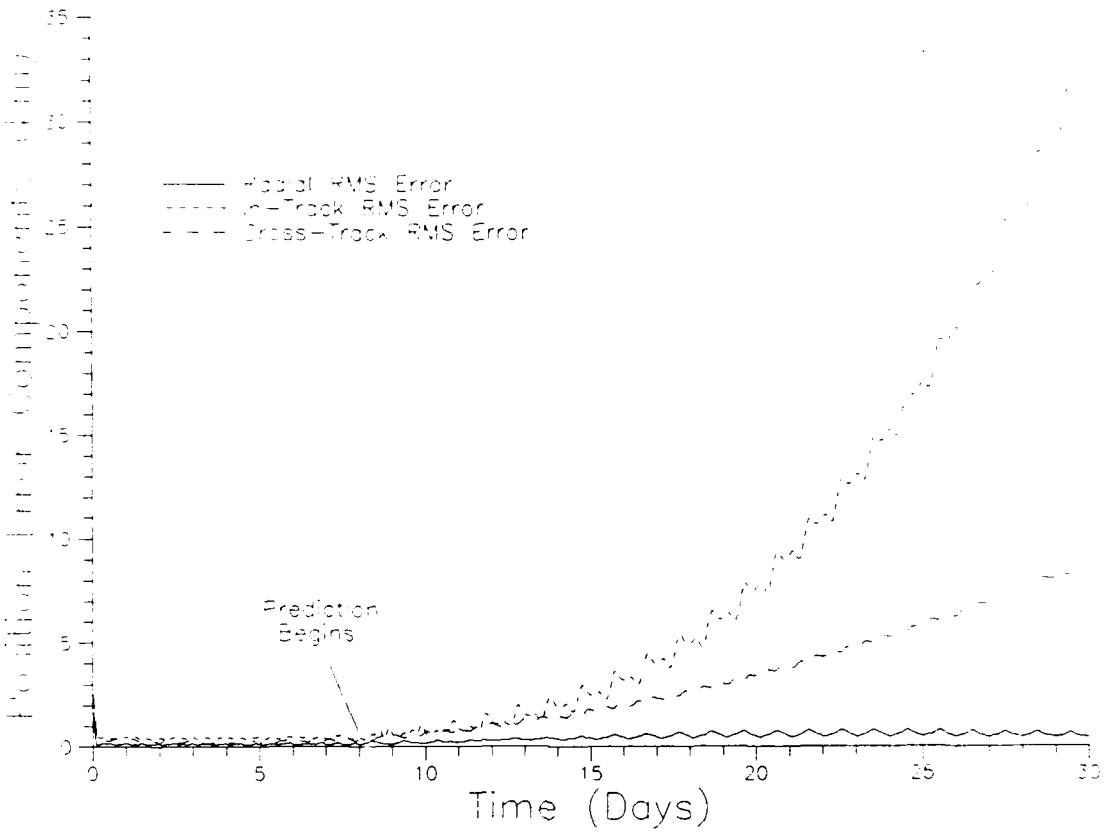

Figure 8-3. Actual (True) RMS Error Components

$\frac{\dot{n}}{2}$ is only an approximation to the effects of drag. The precision will always be limited by the assumption that *all* drag effects can be lumped into this *single* constant. Further, by holding $\frac{\dot{n}}{2}$ constant during the prediction process, the implicit assumption is made that drag effects will remain, on the average, the same. The atmosphere is dynamic enough that there is no reason to believe that even a *perfect* calculation of this value at epoch will remain any more valid than an imperfect estimate when predicting over long times. Indeed, Monte Carlo simulations verified no significant improvements were gained by doubling the data and/or data time span. Therefore, for the purposes of this research,

thirty-five points spread over a week was deemed completely adequate for the two-step algorithm.

The drag parameter was found to be quite observable for this situation. Monte Carlo results produced very similar results between individual runs with the value of $\frac{\dot{n}}{2}$ always lying in the range $2.37 \times 10^{-13} \leq \frac{\dot{n}}{2} \leq 2.39 \times 10^{-13}$ rad/sec^2 with a mean of $\frac{\bar{\dot{n}}}{2} \approx 2.38 \times 10^{-13} \pm 4 \times 10^{-16}$ rad/sec^2 (1σ). The consistency and small standard deviation in these values is further confirmation that the value is quite easily observed after a week with as few as thirty-five points.

An analytic approximation to $\frac{\dot{n}}{2}$ can be calculated to compare with that produced by the Monte Carlo simulations. Differentiating the definition of the mean motion yields:

$$\frac{\dot{n}}{2} = -\frac{3n}{4a}\dot{a} \tag{8-1}$$

It can be readily shown for circular orbits that

$$\dot{a} = \frac{2}{n} F_v \tag{8-2}$$

where F_v is the perturbing acceleration tangent to the orbit (7:399; 8:488). Assuming drag is the only perturbing acceleration, this can be rewritten as:

$$\dot{a} = \frac{2}{n} a_{drag} \tag{8-3}$$

The acceleration due to drag for a circular orbit about a planet with a rotating atmosphere is given by

$$a_{drag} = -\frac{1}{2} \frac{C_D A}{M} \rho v_b^2 \tag{8-4}$$

where the speed of the spacecraft relative to the rotating atmosphere is

$$v_b = \sqrt{(v_x + y\omega)^2 + (v_y - x\omega)^2 + v_z^2} \tag{8-5}$$

and ω is the rotation rate of the planet; C_D is the drag coefficient of the spacecraft; A is the effective spacecraft area; M is the mass, and ρ is the atmospheric density (7:423-424; 28:3-4). v_x, v_y, and v_z are the x, y, and z components of the inertial circular velocity, respectively. Combining Eqs. (8-1), (8-3), and (8-4) produces a simple equation for the drag correction term:

$$\frac{\dot{n}}{2} = \frac{3}{4a} \frac{C_D A}{M} \rho v_b^2 \tag{8-6}$$

Over an orbit, the value of v_b^2 changes as reflected in Eq. (8-5). But $\frac{\dot{n}}{2}$ is assumed to be constant at any point in the orbit, so v_b^2 should be replaced by its orbit-averaged mean value, $\overline{v_b^2}$, in Eq (8-6). This substitution, in effect, is equivalent to averaging both sides of the equation over an orbital period.

$\overline{v_b^2}$ is found by first transforming Eq. (8-5) to its equivalent form in terms of the appropriate equinoctial elements. Assuming all of the elements given in Table 8-1 are approximately constant (slow variables)

except for the mean longitude L, the cartesian position and velocity at any point in this orbit can be represented by

$$\underset{\sim}{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \approx \begin{bmatrix} 0.9415\ \cos L + 0.0585\ \sin L \\ 0.0585\ \cos L + 0.9415\ \sin L \\ -0.3320\ \cos L + 0.3320\ \sin L \end{bmatrix} a \qquad (8\text{-}7a)$$

$$\underset{\sim}{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \approx \begin{bmatrix} 0.0585\ \cos L - 0.9415\ \sin L \\ 0.9415\ \cos L - 0.0585\ \sin L \\ 0.3320\ \cos L + 0.3320\ \sin L \end{bmatrix} v_c \qquad (8\text{-}7b)$$

where $v_c = \sqrt{\frac{\mu}{a}}$ is the inertial circular speed (27:51,73-75). [Note, liberal use of zero eccentricity was employed in obtaining Eqs. (8-7). See Appendix D for details.] Substituting the relations above into v_b^2 and manipulating the algebra produces:

$$v_b^2 \approx v_c^2 - 1.766\ v_c \omega a + \omega^2 a^2 \left(0.8897 + 0.2204\ \sin L \cos L\right) \qquad (8\text{-}8)$$

To average the square of the relative speed over an orbit, the integral

$$\overline{v_b^2} = \frac{1}{2\pi} \int_0^{2\pi} v_b^2\ dL \qquad (8\text{-}9)$$

is computed as

$$v_b^2 \approx v_c^2 \quad 1.766\ v_c \omega a + 0.8897\ \omega^2 a^2 \qquad (8\text{-}10)$$

where it has been further assumed that v_c is constant for the purposes of integration. If drag is ignored in approximating the circular speed, then $v_c \approx \sqrt{\frac{\mu}{a}}$. Recalling $\mu \approx 3.9860045 \times 10^5$ km^3/sec^2 and $\omega \approx 7.29212 \times 10^{-5}$ rad/sec for the Earth (Table A-3) and a = 6785.58 km (Table 8-1), then $v_c \approx 7.664$ km/sec. Eq. (8-10) can be evaluated to yield $\overline{v_b^2} \approx 52.28$ km^2/sec^2.

References cite a wide range of values for the atmospheric densities at the altitude of interest. Values as low as 1.1×10^{-4} kg/km^3 are given for the nighttime atmosphere during periods of low solar activity and as high as 1.3×10^{-2} kg/km^3 for the daytime atmosphere during periods of exceptionally high solar activity (6:221; 26:2-10, 2-11,2-14,2-15; 28:A-4; 37:280; 47:161). Since low-altitude satellites quickly circle the Earth, it is reasonable to assume the orbit-averaged density seen by the satellite will be somewhere between the night and day values. Further, for the purposes of this study, it is appropriate to assume an average solar activity level; therefore a density of $\rho \approx 2.0 \times 10^{-3}$ kg/km^3 is representative of typical conditions at the station's nominal altitude (37:280).

In light of the values above and those in Table 8-1, Eq. (8-6) can be evaluated to produce an analytic approximation for the drag parameter, $\frac{\dot{n}}{2} \approx 2.31 \times 10^{-13}$ rad/sec^2. This value is slightly less than that found in the Monte Carlo study, but well within the accuracy of the approximation. The consistency between the analytic and Monte Carlo values is further corroboration that drag effects are observable when a two-step system is employed.

Predictions Across A Solar Flare-Induced Atmospheric Disturbance

Implicit in the assumption that $\frac{\dot{n}}{2}$ remains constant is the belief that atmospheric density remains essentially constant over the period of prediction. A particularly large solar disturbance can invalidate this assumption for a period of time. A "worst case" arrangement would be for a flare to occur after a spacecraft computes $\frac{\dot{n}}{2}$ and predicts its future position. Because the calculated value becomes invalid, it will underestimate atmospheric drag effects for the duration of the flare's effects on Earth.

Suppose at day seven the satellite wants to predict its position at a later time. Figure 8-4 shows the actual RMS error in this prediction if a solar flare's effects are encountered at day 10.5. For reference, the timing of the solar flare has been plotted below. RMS error values for the equivalent prediction when no flare is encountered are shown with dashed lines for comparison. The atmospheric perturbation causes more than an order of magnitude degradation in precision. While not excellent, predicting positions to 400 km 23 days in advance is still within acceptable limits for many applications.

Since there is no autonomous way for a spacecraft to know when there will be a large disturbance and how long it will last, no apparent solution exists for this scenario. It is possible, however, for the vehicle to determine that such a disturbance has occurred or is occurring. Simply put, the spacecraft can detect such an occurrence when its present position begins to depart from that previously predicted by an unusually large amount; i.e., residual monitoring, etc.

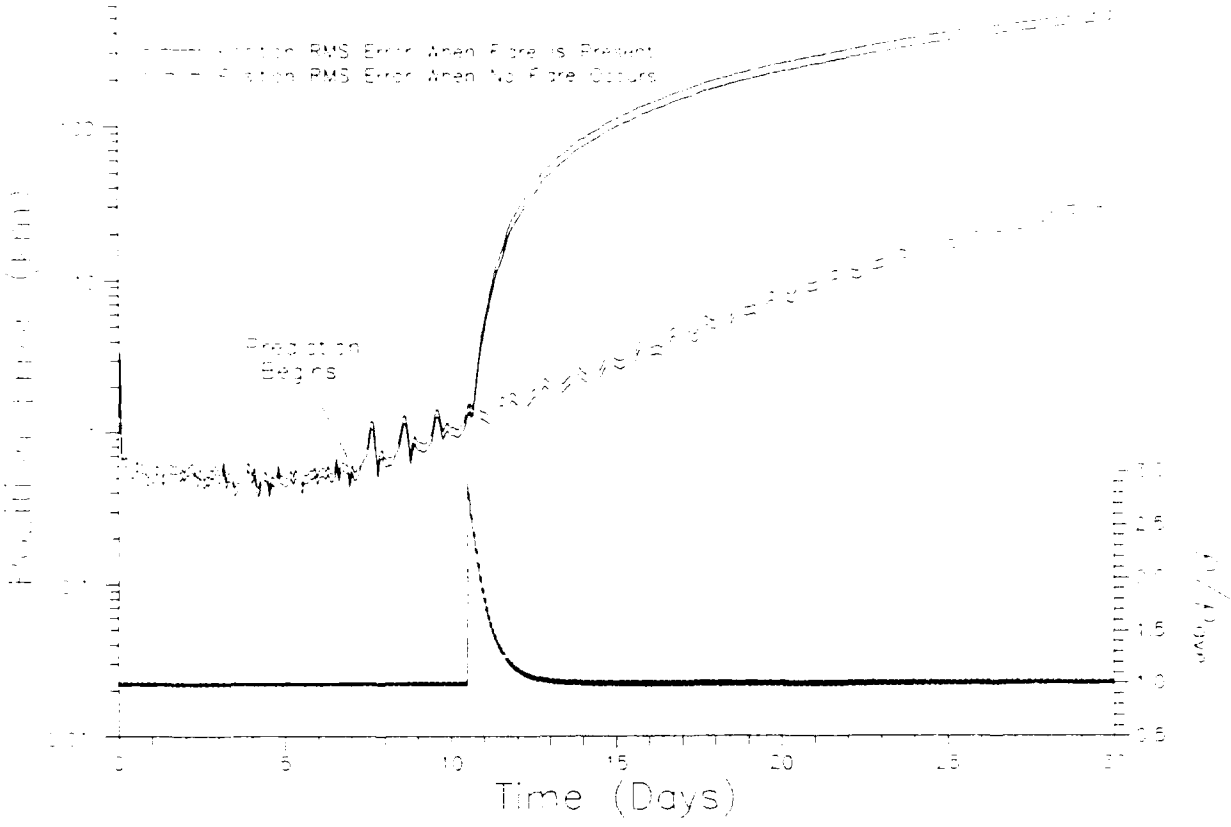

Figure 8-4. Predictions Across a Solar Flare

Once such a determination is made, the spacecraft "suspects" all of its previous predictions beyond the present time to be in error and can take corrective action.

Predictions After A Solar Flare Induced Atmospheric Disturbance

In the last section, the calculation of the drag parameter, $\frac{\dot{n}}{2}$, incorporated data before the disturbance ($1 \leq t \leq 7$ days) and, therefore, underestimated drag effects during the disturbance ($10.5 \leq t \leq 13$ days). The other "worst case" scenario is that an

undetected disturbance occurs during the time just before prediction begins. In this situation, the calculation of $\frac{\dot{n}}{2}$ includes data across the disturbance and the value will overestimate future drag effects.

Figure 8-5 shows the Monte Carlo results. At day 14, the spacecraft predicts its future position. Sixteen days later the predicted position is 800 km in error (1σ). Assuming that the spacecraft has, either autonomously or by human intervention, determined the approximate time span over which the disturbance has occurred, stored data can be manipulated to yield significantly better prediction performance.

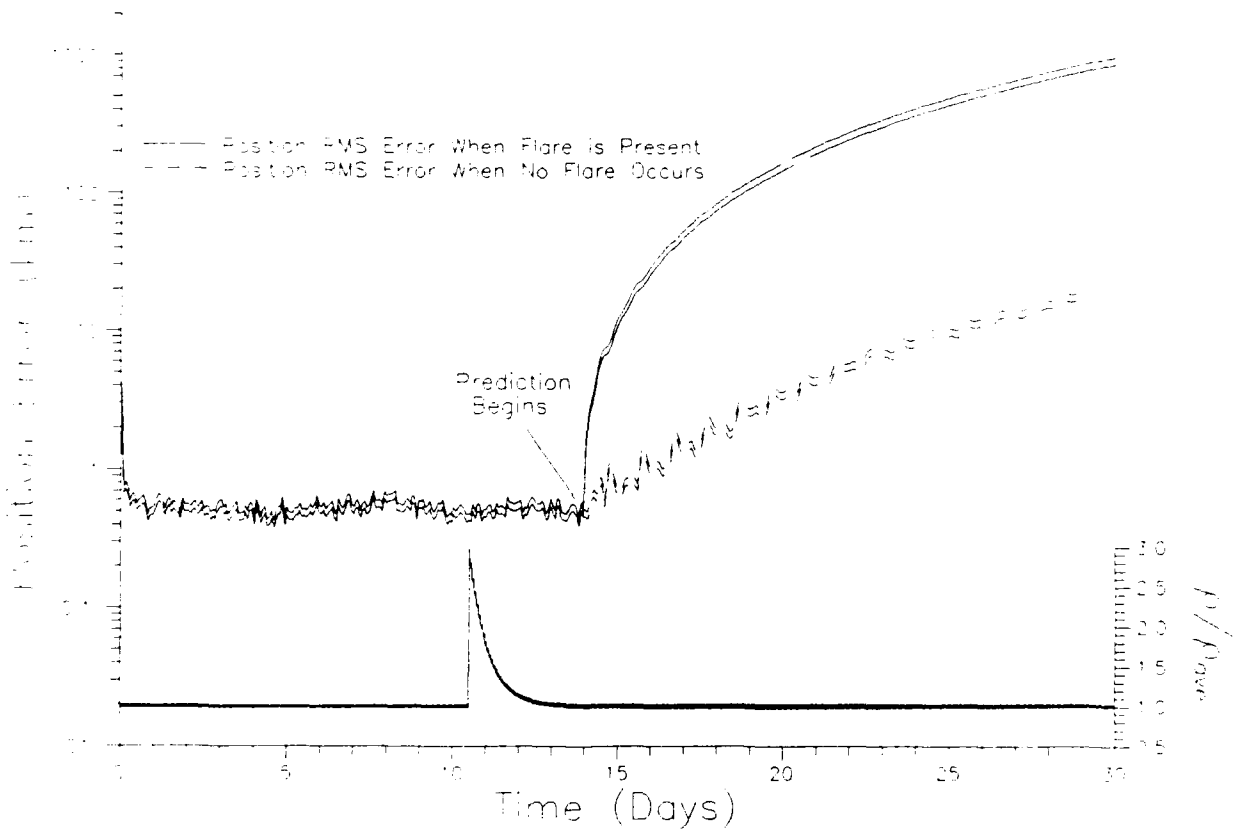

Figure 8-5. Predictions After a Solar Flare

Atmospheric density returns to approximately its nominal value a few days after the disturbance occurs. Further, the spacecraft's semi-major axis will, typically, be reduced only slightly during the disturbance and drag effects experienced afterwards will be comparable to those before. Indeed, in an exponential atmosphere, the density near a reference altitude h_o can be approximated as

$$\frac{\rho_o}{\rho} = e^{-\left[\frac{h_o - h}{H}\right]} \qquad (8\text{-}11)$$

where $H \approx 383$ km is the scale height for this orbit (28:3-4, A-4). So, the density ratio between two altitudes, before and after the disturbance, can be written:

$$\frac{\rho_2}{\rho_1} = e^{-\left[\frac{h_2 - h_1}{H}\right]} \qquad (8\text{-}12)$$

It has been assumed in Eq. (8-12) that $\rho_{o_1} \approx \rho_{o_2}$; i.e., the density at the reference altitude has returned to its preflare value. Eq. (8-12) can be solved to show that a 10% increase in density requires a 36.5 km reduction in the orbit altitude. Therefore, the assumption that drag effects after a disturbance approximate those before is valid even if the semimajor axis decreases by 30 - 40 km during the event.

The calculation of $\frac{\dot{n}}{2}$ from "preflare" data is a straightforward application of methods described in the previous chapter with one exception: the epoch time in the equations is *not* the present time

(t = 14 days); rather, it is the last time before the flare for which the spacecraft has a stored state estimate. Note, then, the resulting epoch corrections, $\delta L_{o_{preflare}}$ and $\delta n_{o_{preflare}}$ are corrections to the preflare epoch and not to the present state. Because of tne unmodelled disturbance between this epoch time and the present, this data contains virtually no information that the onboard system can employ to correct its current state estimate.

"Post-flare" data, conversely, *can* be used to update the current state. Repeating the LLS calculations with data stored from the flare until the present ($13 \leq t \leq 14$ days) generates δL_o, δn_o, and an additional estimate for $\frac{\dot{n}}{2}$. The latter value is discarded. Figure 8-6 graphically summarizes the manner in which data is divided for these calculations as well as the constants retained from each time "slice."

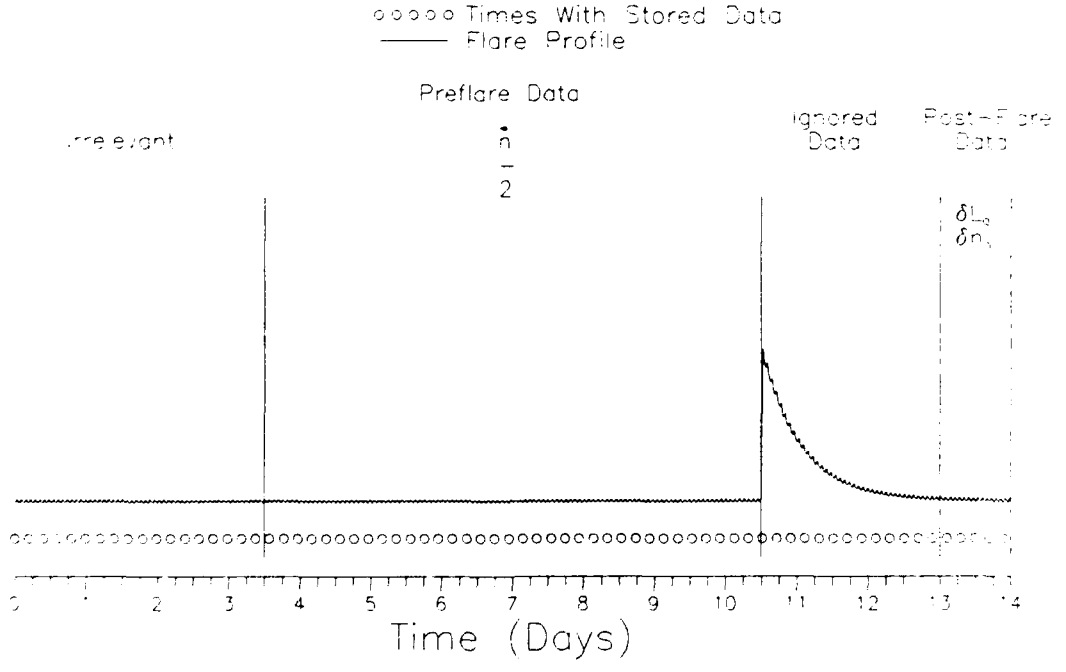

Figure 8-6. The Division of Data to Minimize a Flare's Effect on Prediction Precision

Before presenting results, a few words about the decision to discard the value of $\frac{\dot{n}}{2}$ calculated from post-flare data are in order. The sequence of events studied allowed for only one day of post-flare data to be accumulated before predicting (on day 14). Since drag effects cannot be adequately measured over short times, the previous estimate found with seven days of preflare data will, in general, be the better estimate. As the time between the disturbance and prediction point grows, the post-flare value begins to become more and more certain and the filter can eventually switch back to performing only one LLS operation (as if the flare never occurred).

Figure 8-7 presents the Monte Carlo results obtained for the situation at hand. The RMS error after predicting 16 days has been reduced to about 27 km. Further, the error growth is linear. As Figure 8-8 shows, this is primarily due to a linear growth in the in-track component. This indicates the dominant source of error is due to an error in the epoch ($t = 14$ days) estimate of the semimajor axis and *not* to an inappropriate choice for $\frac{\dot{n}}{2}$.

The figures illustrate much of the accuracy in predictions following a flare can be recovered by breaking the data apart as described. Refining epoch estimates by including more post-flare data would improve prediction capabilities even further. This could be accomplished by increasing the data rate following a solar event or by simply waiting for a longer period of time before attempting to predict future predictions.

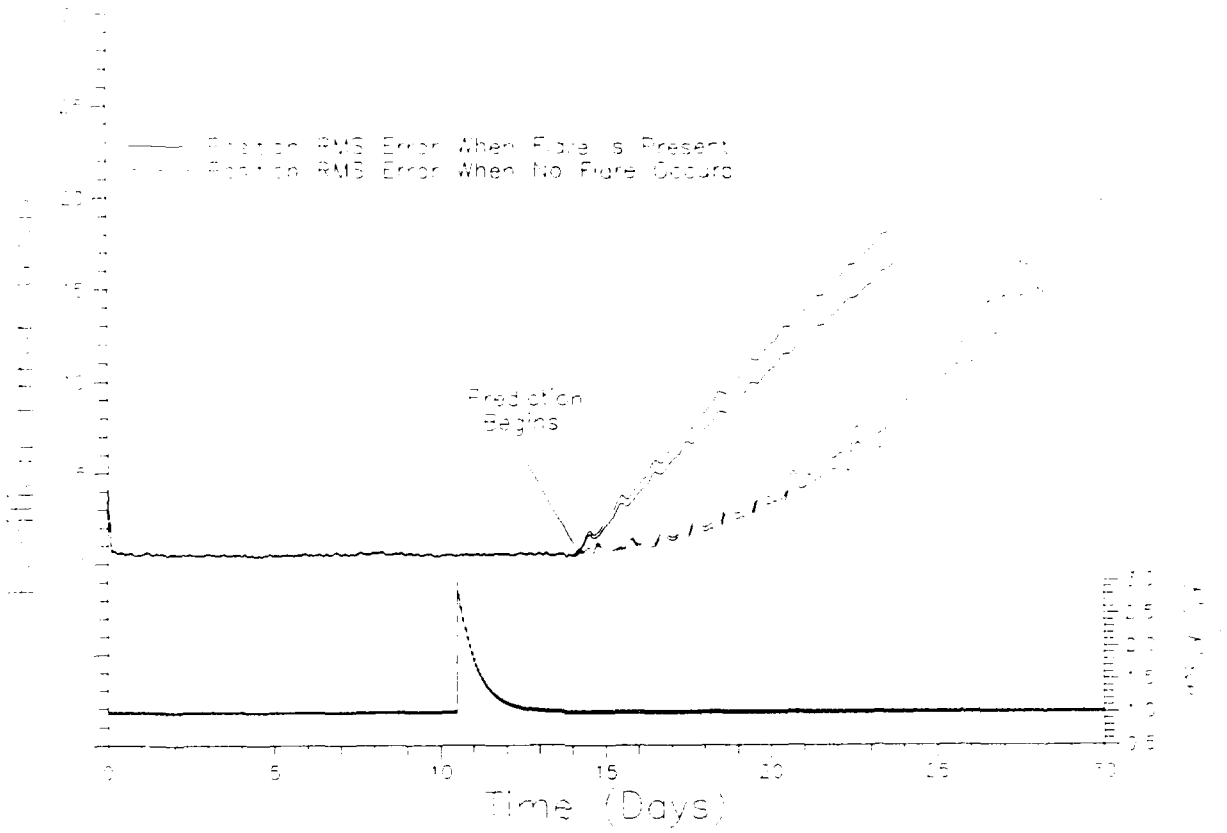

Figure 8-7. Predictions Following a Flare
(With Data Splitting)

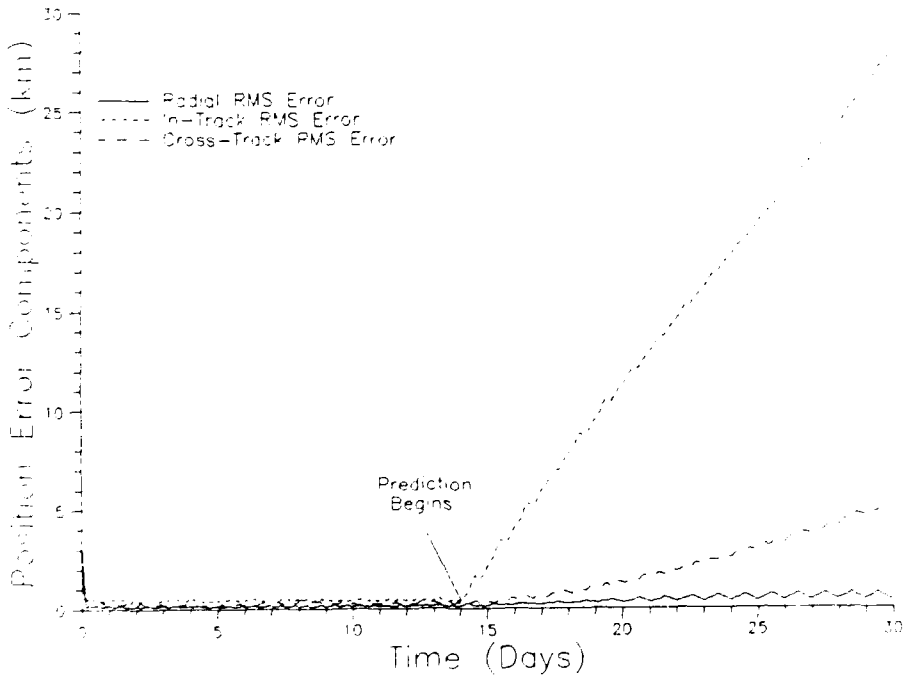

Figure 8-8. Actual (True) Error Components When Predicting After a Flare
(With Data Splitting)

Conclusion

This study of predictions in low-Earth orbit has produced very promising results. Even when applied to a situation where drag is unusually large, prediction capabilities rival those of ground-based facilities under normal operating conditions: 9 - 10 km after two weeks. When solar activity causes unusually large changes in atmospheric density, the method must be modified to some degree, but produces acceptable results, never the less.

IX. Conclusions and Recommendations for Future Study

A truly autonomous navigation system for Earth satellites has been derived and investigated. In addition to estimating a vehicle's current position, the two-step algorithm introduced allows future positions to be predicted accurately even in low-Earth orbit. Major results from this study are summarized below.

System Characteristics

The navigation scheme presented was comprised of existing hardware: an 8088-type microcomputer, star sensors, and Earth sensors. Software included an iterated, extended U-D estimator and a modified form of Aksnes' first-order orbit theory. When applied to an intermediate test orbit, the system was found to exhibit the following characteristics:

1. Subkilometer precision even when observations were as far apart as 100 minutes, allowing a great latitude for time-sharing of the processor with other onboard functions.

2. Subkilometer precision for composite instrument precisions equal to or better than 0.03°, well within current technology limits.

3. Relative insensitivity to instrument bias.

4. Great dependence on star selection for best results. Stars near 22° produced much smaller steady-state position errors.

5. Robust operation. The estimator could be applied to all eccentricities considered and most inclinations without retuning. With proper retuning (when necessary), convergence to steady-state operation was obtained for all inclinations.

Short-Term Estimation Results

The short-term estimation problem was investigated for several useful orbit classes. For low-Earth orbits, subkilometer precision was obtained using the system as derived. To obtain the same precision in semisynchronous orbit, however, modifications to the dynamics model to include third-body perturbations and several sectorial harmonics of the Earth had to be simulated. These modifications are minor and theories exist for them. Synchronous orbits require a little extra care in star selection, but position estimates of less than five kilometers are obtained easily.

Long-Term Estimation Results

Results for Freedom's proposed orbit indicate the two-step method derived greatly enhances the ability to predict future positions in low-Earth orbit -- accuracies rivaling those of ground-based facilities were obtained. Under "normal" conditions, positions could be predicted to ten kilometers two weeks in advance.

Solar flares were shown to degrade the performance when undetected. Once detected, however, appropriate action can be taken, such as discarding predictions across the flare and modifying the calculation of drag effect for use in predictions following the flare. Such a modification resulted in predictions only five kilometers worse than the comparable "no-flare" predictions.

Recommendations for Future Study

With respect to this research, several areas deserve further study. These are briefly summarized in the remaining paragraphs.

1. Different onboard dynamics models need to be investigated. Certain orbits, semisynchronous in particular, require sectorial harmonic terms of the Earth to be modelled onboard for best results. Further, third-body effects of the Sun and Moon should be included for high-altitude orbits. Both general perturbations and numerical integration methods exist for these effects. While this study favored perturbations methods, further study may show numerical integration to be a viable alternative.

2. The system should be applied to *real* data and the estimates compared to the equivalent batch, nonlinear least squares results, an accepted standard for comparison in orbit determination (11:3). While the truth model employed in this simulation was quite accurate, success with real data will convince even the most ardent skeptic.

3. Since many satellites already incorporate the necessary hardware, this scheme could be employed as simply as a software upload (core memory permitting). As a first step, the spacecraft could simply downlink its state estimates for comparison to ground-based calculations. This should be actively pursued as a definitive and final proof of concept.

Appendix A: Assumed Constants in This Research

Table A-1. Earth's Spherical Harmonic Field

n	m	C_{nm}	S_{nm}
2	0	$-0.10826270 \times 10^{-2}$	0.00000000×10^{0}
2	1	0.00000000×10^{0}	0.00000000×10^{0}
2	2	$0.15362188 \times 10^{-5}$	$-0.88149101 \times 10^{-6}$
3	0	$0.25364140 \times 10^{-5}$	0.00000000×10^{0}
3	1	$0.21577626 \times 10^{-5}$	$0.24126717 \times 10^{-6}$
3	2	$0.26584006 \times 10^{-6}$	$-0.25794649 \times 10^{-6}$
3	3	$0.68342574 \times 10^{-7}$	$0.21311125 \times 10^{-6}$
4	0	$0.16233497 \times 10^{-5}$	0.00000000×10^{0}
4	1	$-0.49092463 \times 10^{-6}$	$-0.45669614 \times 10^{-6}$
4	2	$0.76688187 \times 10^{-7}$	$0.15020563 \times 10^{-6}$
4	3	$0.62092126 \times 10^{-7}$	$-0.71253554 \times 10^{-8}$
4	4	$-0.22210654 \times 10^{-8}$	$0.75347615 \times 10^{-8}$
5	0	$0.22608567 \times 10^{-6}$	0.00000000×10^{0}
5	1	$-0.45957673 \times 10^{-7}$	$-0.68484786 \times 10^{-7}$
5	2	$0.96888828 \times 10^{-7}$	$-0.64588236 \times 10^{-7}$
5	3	$-0.19301654 \times 10^{-7}$	$-0.53971560 \times 10^{-8}$
5	4	$-0.90188441 \times 10^{-9}$	$-0.35344245 \times 10^{-9}$
5	5	$0.34362977 \times 10^{-9}$	$-0.21382311 \times 10^{-8}$
6	0	$-0.54248781 \times 10^{-6}$	0.00000000×10^{0}
6	1	$-0.56779905 \times 10^{-7}$	$0.13970346 \times 10^{-7}$
6	2	$0.30690303 \times 10^{-8}$	$-0.50574933 \times 10^{-7}$
6	3	$0.91517316 \times 10^{-10}$	$0.60242316 \times 10^{-9}$
6	4	$-0.37866087 \times 10^{-9}$	$-0.11468848 \times 10^{-8}$
6	5	$-0.10898613 \times 10^{-9}$	$-0.49201945 \times 10^{-9}$
6	6	$-0.67880983 \times 10^{-11}$	$-0.61336581 \times 10^{-10}$

(Adapted from 6:179; 19:Table 18; 28)

Table A-2. Onboard Star Catalog		
Star Name		$\hat{e}_s$ (Geocentric-Equitorial System)
1	β Ursa Minor	$-0.19996\,\hat{e}_x - 0.18443\,\hat{e}_y + 0.96229\,\hat{e}_z$
2	α Andromeda	$+0.87404\,\hat{e}_x + 0.02913\,\hat{e}_y + 0.48497\,\hat{e}_z$
3	α Piscus Anst	$+0.83586\,\hat{e}_x - 0.23637\,\hat{e}_y - 0.49545\,\hat{e}_z$
4	α Taurus	$+0.34720\,\hat{e}_x + 0.89385\,\hat{e}_y + 0.28369\,\hat{e}_z$
5	α Carina	$-0.06239\,\hat{e}_x + 0.60294\,\hat{e}_y - 0.79534\,\hat{e}_z$
6	α Gemini	$-0.33732\,\hat{e}_x + 0.77887\,\hat{e}_y + 0.52875\,\hat{e}_z$
7	α Canis Major (Sirus)	$-0.18485\,\hat{e}_x + 0.93984\,\hat{e}_y - 0.28728\,\hat{e}_z$
8	α Leo	$-0.86275\,\hat{e}_x + 0.46061\,\hat{e}_y + 0.20857\,\hat{e}_z$
9	α Virgo	$-0.91550\,\hat{e}_x - 0.35341\,\hat{e}_y - 0.19228\,\hat{e}_z$
10	α Bootes	$-0.78497\,\hat{e}_x - 0.52445\,\hat{e}_y + 0.32982\,\hat{e}_z$
11	α Scorpios (Antares)	$-0.34814\,\hat{e}_x - 0.82528\,\hat{e}_y - 0.44465\,\hat{e}_z$
12	α Lyra (Vega)	$+0.12345\,\hat{e}_x - 0.76983\,\hat{e}_y + 0.62619\,\hat{e}_z$
13	α Grus	$+0.60085\,\hat{e}_x - 0.32179\,\hat{e}_y - 0.73173\,\hat{e}_z$
14	α^1 Crux	$-0.45082\,\hat{e}_x - 0.05094\,\hat{e}_y - 0.89116\,\hat{e}_z$

(Adapted from 41:H1-H31)

Table A-3. Various Other Parameters	
Products of gravitational constant and body mass:	
	$\mu_{Earth} = 3.9860045 \times 10^{5}$ km^3/sec^2
	$\mu_{Sun} = 1.3271244 \times 10^{11}$ km^3/sec^2
	$\mu_{Moon} = 4.9027900 \times 10^{3}$ km^3/sec^2
Radius of Earth:	$r_{Earth} = 6378.14$ km
Rotation Rate of Earth:	$\omega = 4.178074216 \times 10^{-3}$ deg/sec
Ellipticity of Earth:	$\epsilon = 8.182 \times 10^{-2}$

(From Ref. 28)

APPENDIX B: PROGRAM LISTING

The source code for the iterated, extended U-D filter employed in this research begins on the next page. The version listed contains many extra calculations and print statements which have been "commented out." Such statements were used in debugging the code and have been left in for future reference. Comments in the code should be sufficient for the interested reader to follow the logic.

```
      program udfil58
c
c
c This uses a two step process to get 3 constants.  First the
c epoch semimajor axis is corrected using delta a and then all
c data is used to get the 3 constants.
c
c This version finds 3 independent correction terms for
c use in the prediction section.  The mean longitude correction
c is generated by assuming an error of the form:  at^2 + bt + c,
c where a is the ndot over 2 correction for drag, b is the linear
c correction due to an error in the estimated semimajor axis at
c epoch and c is the correction to mean longitude at epoch.  The
c semimajor axis error is assumed to be of the form:  dt + e, where
c d is the linear decrease in the semimajor axis due to the
c effect of drag and e is the correction to the semimajor axis
c at epoch.  Note that there are only three independent constants
c here because:  a=a(d) and b=b(e).
c
c The calculation of a (and d) is based on 34 stored values, spread
c over the last week and the current state estimate.  Thus, data is
c 35 delta semimajor axis values and 35 delta mean longitude values.
c
c This version predicts into the future after the data in the input
c file runs out.  Consequently, an extra input variable is required
c -- endtime.  endtime tells the program what time to predict until.
c
c This version allows points to *not* converge.  The maximum number of
c "bomb points" is set by assuming they all occur one after another
c and we only allow this to continue for 1/4 the orbital period or
c ten points, whichever is longer.  Note that this will be a wasted
c effort if the solution is truely diverging from the correct state
c We would get maxbomb failures one after another; however, if the
c nonconvergence is due to the iterations oscillating about the true
c state, then this method has hope.
c
c When the filter fails, a file is written to indicate that fact.
c
c INPUT from local directory and default input
c OUTPUT to /tmp2/khicks directory
c
c U(-) and D(-) are read in in terms of the refel.  The data-
c generating program performs the change from the known P(-)
c in terms of randv to U(-) and D(-) in terms of reference elements.
c Further, xhat (initial elements) is read in directly....this is
c yet another function of the data-generating program.
c
c This program numbers the output files with the case # it reads
c from the first line of the input file.
c
c The P(-)'s are only calculated for refel...the lines computing
c P(-) for randv and rvh systems have been commented out.
c
c To save disk space, the only time information sent to
```

```
c the output files is deltat.  Futher, all output except
c randv information isn't printed.
c Output information about the filter performance is reduced to
c only information essential to the study of the filter performance.
c (Many of the other calculations/outputs are still in the program, but
c have been commented out.)
c
c
c Let's define some of the more important variables:
c
c u, d, pminus -- the UD factors of pminus (the covariance
c                 before incorporting a data point) in terms
c                 of the equinoctal orbital elements.
c
c uplus, dplus, pplus -- ditto above, but after updacing.
c
c xhat -- the current state...both before and after update.
c         1 -- a sub f
c         2 -- a sub g
c         3 -- semimajor axis
c         4 -- mean longitude
c         5 -- chi
c         6 -- psi
c              (as defined on page 68 of Aeronutronic Publication
c              No. U-4180).
c
c randv -- 1-3 are x, y, z and 4-6 are xdot, ydot, zdot --
c          position and velocity corresponding to xhat (or xref).
c
c xref -- state vector about which we linearize.  After
c         the update iteration scheme, it is a carbon copy
c         of the updated state.
c
c deltax -- differential correction to state in the state
c           update iteration scheme.
c
c xnew -- state vector used to hold a new estimate in the
c         iteration scheme.
c
c caph -- linearized observation matrix "capital H" in terms
c         of the state variables.  It is partial(obs)/partial(xhat).
c
c h -- expected observation based on current elements.
c
c capr -- rotation matrix "capital R" between state and randv
c         coordinate system.  R=partial(randv)/partial(elements).
c
c prandv -- covariance in terms of randv.
c
c rmag, vmag, hmag -- magnitude of radial, in-track, and
c                     cross track vectors.
c
c prvh -- covariance for above variables.
c
```

```
c oldxhat, oldpplus -- stored states and covariances
c                      to be used in getting drag factors.
c
c mldiff. mlsig2 -- differences in actual mean longitude
c                   and predicted w/ corresponding variances
c
c adiff,asig2 -- ditto above for semimajor axis
c
c time -- the times (relative) that the above data was
c         recorded.
c
c t -- absolute time from beginning of data taking.
c
c deltat -- time between observations.
c
c endtime -- the time to which to predict if we run out
c            of incoming data.  Set to zero if no prediction
c            desired.
c
c tlast  -- absolute time of the last point stored in
c           oldxhat.
c
c reftime -- last time with data; the point from which
c            prediction occurs.
c
c dt, dt2, dt3, dt4 -- various powers of time used here
c                      and there.
c
c stemp, stemp2, stmpt2 -- scalar temp variables.
c
c vtemp -- temp vector.
c
c mtemp -- temp matrix.
c
c ndoto2 -- the "constant" used to account for drag as calculated
c           from the delta a data -- this is just an interesting
c           variable I watched -- not used for anything.
c
c coeff -- constants used in fitting the errors in state
c          to account for drag.  Elements are:  Delta mean longitude
c          (L) (or, equivalently, mean anomaly (M)) at epoch, delta
c          mean motion (n) at epoch, and n dot over 2.  This array is
c          also used in the semimajor axis correction step (an optional
c          step as described in the text.)
c
c pcoeff -- covariance corresponding to coeff.
c
c sum1, sum2 -- used in estimating coeff.
c
c eradius -- Earth's radius in distance units.
c
c eterms -- GM (DU**3/TU**2), J2, J3, J4, J5 for the Earth.
c
c star  -- unit vector to current star in geocentric-
```

```
c         equatorial coordinate system.
c
c starvec -- the catalog of star vectors.
c
c indx -- used by two subroutines to pass information back
c         and forth.
c
c repeat -- logical variable that tells us when to continue
c           iterating.
c
c filename, case, digit -- the makings of the output file names.
c
c n -- order of the state.
c
c numit -- iteration number.
c
c maxit -- maximum number of iterations.
c
c ibombed -- # of times convergence failed.
c
c istored -- # of points stored in oldxhat.
c
c istar -- star number of current observation.
c
c ibetween -- number of data points between storing values
c             in oldxhat.
c
c The following correspond to terms used in the update and
c propagation section.  Page numbers correspond to those in
c Maybeck, Volume 1:
c a -- the scalar a in kalman update (p 393).
c z -- the observation (scalar).
c q -- covariance (scalar) of the observation.
c qn, cn -- factored propagation noise matrix in terms of state
c           variables (equinoctal elements).  These are Gd and Qd
c           in Maybeck.)
c avec -- the "a" vector (p 394).
c kg -- kalman gain vector.
c f -- used in update (p 394).
c b -- ditto above.
c v -- ditto above.
c p -- ditto above.
c dtilda -- "D super tilda" in propagation of
c           covariance (p 396).
c y -- "Capital Y" matrix in prop. of cov. (p 396).
c phi -- state transition matrix.
c
c
      double precision u(6,6),d(6),z,q,qn(6),cn(6,6),xhat(6),
     1  dplus(6),uplus(6,6),avec(6),stemp,stemp2,stmpt2,vtemp(6),
     2  mtemp(6,6),kg(6),a,caph(6),h,deltax(6),xref(6),f(6),b(6),
     3  v(6),p(6),dtilda(12),y(6,12),phi(6,6),pminus(6,6),t,deltat,
     4  endtime,star(3),randv(6),starvec(14,3),pplus(6,6),
     5  sigma2,capr(6,6),prandv(6,6),xnew(6),prvh(3,3),rmag,vmag,hmag
```

```
      double precision oldxhat(6,35),oldpplus(6,6,35),dt,dt2,dt3,dt4,
     1  reftime,tlast,mldiff(35),mlsig2(35),time(35),ndo2sig2,
     2  ndoto2,eradius,eterms(5),adiff(35),asig2(35),suml(3,3),
     3  sum2(3),pcoeff(3,3),coeff(3)
      integer indx(3)
      logical repeat
      character*1 digit(10)
      character*6 case
      character*8 failname
      character*19 file9
      character*20 file7
      character*21 file3,file5
      character*22 file2,file4
      common /fudge/ coeff
      common /earth/ eradius,eterms
      data digit /'0','1','2','3','4','5','6','7','8','9'/
c
      case(1:4)='case'
      file2(1:20)='/tmp2/khicks/prefel.'
      file3(1:19)='/tmp2/khicks/refel.'
      file4(1:20)='/tmp2/khicks/prandv.'
      file5(1:19)='/tmp2/khicks/randv.'
      file7(1:18)='/tmp2/khicks/prvh.'
      file9(1:17)='/tmp2/khicks/rvh.'
      failname(3:8)='failed'
c
 1501 format (3(1x,e23.16))
 1502 format (2x,a)
 1503 format (a)
 1504 format (1x,f12.4,1x,i2,1x,e23.16,1x,i2)
 1505 format (5(1x,e23.16))
c
      read (*,*) istart,istop
c Open the input file and the the case number to start
c things off....
      do 9999 icase=istart,istop
c set up the input file name:
        n1=icase/10
        n2=icase-n1*10
        case(5:5)=digit(n1+1)
        case(6:6)=digit(n2+1)
        write (*,*) 'opening ',case
        open (8,file=case,status='old')
c Read and ignore the case name....
        read (8,1502) case
c
c set up the output file names:
      file7(21:22)=case(5:6)
      file3(20:21)=case(5:6)
      file4(21:22)=case(5:6)
      file5(20:21)=case(5:6)
      file7(19:20)=case(5:6)
      file9(18:19)=case(5:6)
c
```

```
c     Open the other I/O files:
c     open (2,file=file2)
c     open (3,file=file3)
      open (4,file=file4)
      open (5,file=file5)
c     open (7,file=file7)
c     open (9,file=file9)
      open (10,file='star.dat',status='old')
c
c     Get the star vectors from 'star.dat'
      do 2000 i=1,14
         read (10,*) istar,(starvec(i,j),j=1,3)
 2000 continue
c
c     Set up the Earth constants....
      call terra
c
c     Now set some internal constants (like turning off drag):
      n=6
      t=0.0d+00
      maxit=3
      ibombed=0
      istored=0
      do 3510 i=1,3
         coeff(i)=0.0d+00
 3510 continue
c
c GET THE INPUT INFORMATION
c     First, get U(-) for refel:
      do 1001 i=1,6
         read (8,*) (u(i,j),j=1,6)
 1001 continue
c     Now get D(-) for refel:
      read (8,*) (d(j),j=1,6)
c     Get Cn (identity matrix) for refel:
      do 1002 i=1,6
        read (8,*) (cn(i,j),j=1,6)
 1002 continue
c     Qn for refel:
      read (8,*) (qn(i),i=1
c     Estimate of state at time zero:
      read (8,*) (xhat(i),i=1,6)
c     Sigma squared of the instrument:
      read (8,*) sigma2
c     Time between observations:
      read (8,*) deltat
c read in "predict until" time:
      read (8,*) endtime
c
c set up the integer number of points between mean longitude
c storages (store 5 points per day):
      ibetween=21.4/deltat
c
c set up the maximum number of nonconvergent points (1/4 period
```

```
c or 10, whichever is more):
      maxbomb=.25*(6.2832/sqrt(eterms(1)/xhat(3)**3))/deltat
      if (maxbomb .lt. 10) maxbomb=10
c
c echo it back to the default output to insure proper read
      write (*,*) 'U(-) for refel:'
      do 1003 i=1,6
        write (*,1501) (u(i,j),j=1,6)
 1003 continue
      write (*,*) 'D(-) for refel:'
      write (*,1501) (d(j),j=1,6)
      write (*,*) 'Cn for reference elements:'
      do 1004 i=1,6
        write (*,1501) (cn(i,j),j=1,6)
 1004 continue
      write (*,*) 'Qn for reference elements:'
      write (*,1501) (qn(i),i=1,6)
      write (*,*) 'xhat:'
      write (*,1501) xhat
      write (*,*) 'sigma squared of the instrument:',sigma2
      write (*,*) 'delta t=',deltat
      write (*,*) 'Predict until ',endtime,' when data runs out.'
      write (*,*) 'Allow ',maxbomb, 'points to not converge in '
     1  ,maxit,' iterations.'
c Get the Pminus matrix for the reference elements...this is
c just to monitor the initialization of algorithm.
      call formp(u,d,pminus)
      write (*,*) 'Pminus for refel:'
      do 312 i=1,6
         write (*,1501) (pminus(i,j),j=1,6)
  312 continue
      write (*,*)
c
c We can hardwire some of the covariance propagation algorithm
c (Maybeck, page 397):
      do 1005 i=1,n
        do 1006 j=1,n
           y(i,j+n)=cn(i,j)
 1006   continue
        dtilda(i+n)=qn(i)
 1005 continue
c
c Now to initialize the theory......
      call esat6(0.0d+00,xhat,randv,capr,1,0)
c
c Put the deltat at the top of all of the output files:
c     write (2,*) deltat
c     write (3,*) deltat
      write (4,*) deltat
      write (5,*) deltat
c     write (7,*) deltat
c     write (9,*) deltat
c
c
```

```
c     HERE'S THE MAIN OBSERVATION LOOP:
 1000 continue
      do 3000 mloop=1,ibetween
c     first point read is at time zero.....
        read (8,*,end=935) istar,z
        do 2010 i=1,3
           star(i)=starvec(istar,i)
 2010   continue
c At this point, we have xhat(-),D(-),U(-),z,Q,Qn,Cn,CapR
c
c Form P(-) for numerical efficiency....
      call formp(u,d,pminus)
c
c ***************************************************************************
c *    These next calculations and prints of P(-) were for                  *
c *    debugging.....left as comments in case they are needed               *
c *    to study the filter performance later.                               *
c ***************************************************************************
c Need to get P(-) for randv while we're at it.. .
c First, we'll need capr for the rotation:
c     call esat6(0.0,xhat,randv,capr,1,1)
c     call convp(pminus,prandv,capr,6,6)
c Get covariance for rvh frame
c     call rot2rvh(randv,prandv,rmag,vmag,hmag,prvh)
c Now write the diagonal elements of the P(-)'s:
c     write (2,1501) (pminus(jj,jj),jj=1,6)
c     write (4,1501) (prandv(jj,jj),jj=1,6)
c     write (7,1501) (prvh(jj,jj),jj=1,3)
c While we're at it, let's print our preupdated values (-):
c     write (3,1501) xhat
c     write (5,1501) randv
c     write (9,1501) rmag,vmag,hmag
c
c Set up initial reference state....
      do 30 i=1,n
         xref(i)=xhat(i)
   30 continue
      numit=1
c
c
c Next comes the iteration loop.......
   35 repeat=.false.
c Set up the initial caph vector and get h (expected observation).
c Capr comes along for the ride.....
      call linearh(xref,randv,star,caph,capr,h)
      q=sigma2*(1.d+00-h*h)
c
c The next bit basically follows notation on 393 of Maybeck,
c Volume 1.
c Evaluate the scalar "a"
      do 40 j=1,n
        vtemp(j)=0.0d+00
        do 45 k=1,n
          vtemp(j)=vtemp(j)+caph(k)*pminus(k,j)
```

```
   45   continue
   40 continue
      a=0.0d+00
      do 50 j=1,n
        a=a+vtemp(j)*caph(j)
   50 continue
      a=a+q
c Evaluate Kalman Gain vector (kg)
      do 55 i=1,n
        kg(i)=0.0d+00
        do 60 k=1,n
          kg(i)=kg(i)+pminus(i,k)*caph(k)
   60   continue
        kg(i)=kg(i)/a
   55 continue
c Update reference trajectory.....
      stemp=0.0d+00
      do 65 j=1,n
        stemp=stemp+caph(j)*(xhat(j)-xref(j))
   65 continue
      stemp=z-h-stemp
      do 70 j=1,n
        deltax(j)=kg(j)*stemp
        xnew(j)=xhat(j)+deltax(j)
c simultaneously check the convergence:
        if (dabs(xref(j)-xnew(j)) .gt. .1d+00*dsqrt(dabs(
     1  pminus(j,j)))) repeat=.true.
c now copy that new element to xref in case we need to repeat!
        xref(j)=xnew(j)
   70 continue
c
c
c At this point, you'd expect to calculate a new randv corresponding
c to the new state.  This is done in a hidden manner -- the next call
c to linearh will return a revised randv as a byproduct!
c
c check for repeat
      if (repeat) then
        if (numit .ge. maxit) then
          if (ibombed .ge. maxbomb) then
            write (*,*) 'Aborted after ',ibombed,'points failed to '
            write (*,*) 'converge in ',maxit, 'iterations.'
            failname(1:1)=digit(n1+1)
            failname(2:2)=digit(n2+1)
            open (11,file=failname)
            write (11,*) 'Failed due to number of bombed iterations.'
            close(11)
            goto 935
          else
c           We can't seem to converge...let's just propagate xhat(-)
c           forward and pray for the best.  As long as we were just
c           oscillating about the solution, this is acceptable,
c           but if we were diverging from the solution, all hell will
c           break loose at the next few points.  In other words, if the
```

```
c           filter really is failing, then we're just delaying the process
c           until a few more points have been included in the estimate.
c           Also, create a VALID xref in case we go directly into the
c           calculation of drag parameters.
            write (*,*) 'failed to converge at',t
            ibombed=ibombed+1
            do 5921 i=1,6
              dplus(i)=d(i)
              xref(i)=xhat(i)
              do 5922 j=1,6
                uplus(i,j)=u(i,j)
 5922         continue
 5921       continue
c           numit=0
c           write (*,1504) t,istar,z,numit
c           since we're calling the t- stuff the t+ stuff, skip over the
c           update section to the propagation section....as a result we'll
c           need capr and to replace randv w/ the values for xhat(-)
            call linearh(xhat,randv,star,caph,capr,h)
            goto 5923
          endif
        else
          numit=numit+1
          goto 35
        endif
      else
c       write (*,1504) t,istar,z,numit
      endif
c Now we have xhat(ti+) (the updated state)....
      do 80 j=1,n
        xhat(j)=xref(j)
   80 continue
c Revise estimate of caph, and randv.  An added bonus is
c that this will bring back capr for the conversion of P(+)
c (if we're going to compute it):
      call linearh(xhat,randv,star,caph,capr,h)
c
c Update the covariance.  Use Maybeck, page 394:
c
      do 110 i=1,n
         f(i)=0.0d+00
         do 115 k=1,n
            f(i)=f(i)+u(k,i)*caph(k)
  115    continue
         v(i)=d(i)*f(i)
  110 continue
      avec(1)=q+f(1)*v(1)
      dplus(1)=d(1)*q/avec(1)
      b(1)=v(1)
      do 120 k=2,n
         avec(k)=avec(k-1)+f(k)*v(k)
         dplus(k)=d(k)*avec(k-1)/avec(k)
         b(k)=v(k)
         p(k)=-f(k)/avec(k-1)
```

```
            do 125 j=1,k-1
               uplus(j,k)=u(j,k)+b(j)*p(k)
               b(j)=b(j)+u(j,k)*v(k)
  125       continue
  120    continue
c Now, since uplus is upper unitary, put the ones on the diagonal....
         do 130 i=1,n
           uplus(i,i)=1.0d+00
  130    continue
c
 5923    continue
c        The above continue (5923) is where we'd jump to if we skipped an
c        update due to nonconvergence.  Note that the covariance matrix is
c        going to grow as a result of ignoring data if we did jump to here!
c
c        Write our updated variables (+) as we get them..
c        write (3,1501) xhat
c        Last call to linearh got randv for xhat(+) so:
         write (5,1501) randv
c
c Form P(ti+) and print it out....just for watching...filter doesn't
c need it!
c this is UD:
         call formp(uplus,dplus,pplus)
c
c Still have capr matrix from the last call to linearh, so
c convert and print the diagonal elements:
         call convp(pplus,prandv,capr,6,6)
c        call rot2rvh(randv,prandv,rmag,vmag,hmag,prvh)
c        write (2,1501) (pplus(jj,jj),jj=1,6)
         write (4,1501) (prandv(jj,jj),jj=1,6)
c        write (7,1501) (prvh(jj,jj),jj=1,3)
c
c We still need to write rmag,vmag, and hmag (+):
c        write (9,1501) rmag,vmag,hmag
c
c End of Updating!..............
c
c Now we propagate the state forward and get phi matrix
         call esat6(deltat,xhat,randv,phi,4,1)
c
c Propagate covariance to next observation time.............
         call propcov(y,dtilda,cn,phi,uplus,dplus,n,u,d)
c
c Now we have propagated the covariance P(-) at next time.....
c
c That's the end of this observation......loop back to get next obs!
         t=t+deltat
 3000    continue
c
c Okay, its time to store some values in case we decide to predict:
         if (istored .eq. 34) then
c          We've stored enough data, so let's shift the data to keep only
c          the most recent values.
```

```
c       Save the last time stored:
        tlast=t-deltat
c       Shift the vector down:
        do 1103 i=2,34
          do 1104 j=1,6
            oldxhat(j,i-1)=oldxhat(j,i)
            do 1108 k=1,6
               oldpplus(j,k,i-1)=oldpplus(j,k,i)
 1108       continue
 1104     continue
 1103   continue
c       Now to store the newest stuff...recall that we've got the
c       updated state before the last propagation stored as xref
c       because we never overwrote that vector.  Also, we have the
c       makings of P(+) at that time still around.
c       The next line would be necessary if we had yet to assemple
c       P(+):
c       call formp (uplus,dplus,pplus)
        do 1109 i=1,6
           oldxhat(i,34)=xref(i)
           do 1110 j=1,6
             oldpplus(i,j,34)=pplus(i,j)
 1110      continue
 1109   continue
c
      else
c
c       We don't have enough points to average yet -- we want to wait
c       one week.  So, tack this one into our arrays:
        istored=istored+1
c       If we didn't have pplus already assembled, we'd need the next line:
c       call formp(uplus,dplus,pplus)
c       Store this stuff at the end of the array:
        do 1120 i=1,6
           oldxhat(i,istored)=xref(i)
           do 1121 j=1,6
              oldpplus(i,j,istored)=pplus(i,j)
 1121      continue
 1120   continue
      endif
      goto 1000
  935 write (*,*)
      t=t-deltat
      reftime=t
      write (*,*) 'Last time with data was:  ',reftime
      write (*,*) 'Convergence Failed at:', ibombed, 'points.'
      write (*,*)
      write (*,*) 'Finished with input for ',case
      write (*,*) 'Beginning prediction until ',endtime
c
c     Begin the prediction section.....
c
c     Now let's take the current point and propagate backwards in
c     time to get the predicted mean longitude w/o drag and the
```

```
c     predicted semimajor axis values.  At the same time, get
c     estimates as to the accuracy (the sigma squareds) of the
c     estimates to incorporate into a weighted least squares routine.
c     Before propagation things, note that the current time can
c     generate a data point also.  This is a zero difference between
c     the mean longitude w/ and w/o drag [stored xhat(+) minus pre-
c     dicted xhat(-)] and a zero difference between estimated (w/
c     drag) and predicted (w/o drag) semimajor axis.
c     The following line would be necessary if we had yet to assemble
c     P(+):
c     call formp(uplus,dplus,pplus)
c     Now for the data from current point:
      time(1)=0.0d+00
      mldiff(1)=0.0d+00
      mlsig2(1)=2.0d+00*pplus(4,4)
      adiff(1)=0.0d+00
      asig2(1)=2.0d+00*pplus(3,3)
c
c     Now for the stored points....handle the most recent one (at
c     t=tlast) separately.  But, first, recall that the updated state
c     and covariance for the most recent time with data is still
c     available as stored in xref, uplus, and dplus!  While we're at it,
c     let's tell the user what those values were.
      write (*,*)
      write (*,*) 'The last state with data is estimated as:'
      write (*,1501) xref
c
c
c     This is the optional step that corrects the initial estimate of
c     the epoch semimajor axis:
c     Since a call to esat6 with k=4 (or 5) screws up the calling argue-
c     ments, call with a copy of the epoch state:
      do 3600 j=1,6
        vtemp(j)=xref(j)
 3600 continue
c     Create data for last stored point...
      dt=-(t-tlast)
c     Get phi:
      call esat6(dt,vtemp,randv,phi,4,1)
c     Propagate covariance using Eq. 4-10 if weighted least squares
c     is desired.  (pseudonoise matrix=0 for determanistic portion)
      call convp(pplus,mtemp,phi,6,6)
      time(2)=dt
      adiff(2)=oldxhat(3,34)-vtemp(3)
      asig2(2)=oldpplus(3,3,34)+mtemp(3,3)
c     The rest of the stored points are evenly spaced, so a do loop is
c     perfect...
      do 3610 i=33,1,-1
      dt=dt-deltat*ibetween
      do 3620 j=1,6
         vtemp(j)=xref(j)
 3620 continue
      call esat6(dt,vtemp,randv,phi,4,1)
      call convp(pplus,mtemp,phi,6,6)
```

```
      time(36-i)=dt
      adiff(36-i)=oldxhat(3,i)-vtemp(3)
      asig2(36-i)=oldpplus(3,3,i)+mtemp(3,3)
 3610 continue
c     Zero out the sums:
      do 3611 i=1,2
        do 3612 j=1,2
          suml(i,j)=0.0d+00
 3612   continue
        sum2(i)=0.0d+00
 3611 continue
c     First the terms from delta semimajor axis:
      write (*,*) 'The data for delta semimajor axis is:'
      do 4113 i=1,35
c       we decided against weighted least squares so this next line
c       is necessary:
        asig2(i)=1.0d+00
        write (*,1501) time(i),adiff(i),asig2(i)
        dt=time(i)
        dt2=dt*dt
        suml(1,1)=suml(1,1)+1.0d+00/asig2(i)
        suml(1,2)=suml(1,2)+dt/asig2(i)
        suml(2,1)=suml(2,1)+dt/asig2(i)
        suml(2,2)=suml(2,2)+dt2/asig2(i)
        sum2(1)=sum2(1)+adiff(i)/asig2(i)
        sum2(2)=sum2(2)+adiff(i)*dt/asig2(i)
 4113 continue
      pcoeff(1,1)=1.0d+00
      pcoeff(1,2)=0.0d+00
      pcoeff(2,1)=0.0d+00
      pcoeff(2,2)=1.0d+00
      call ludcmp(suml,2,3,indx,stemp)
      do 3630 j=1,2
          call lubksb(suml,2,3,indx,pcoeff(1,j))
 3630 continue
      do 3640 i=1,2
         coeff(i)=0.0d+00
         do 3650 j=1,2
            coeff(i)=coeff(i)+pcoeff(i,j)*sum2(j)
 3650    continue
 3640 continue
      write (*,*)
      write (*,*) 'the coeff are'
      write (*,1505) coeff(1), coeff(2)
      write (*,*) 'with covariance of'
      write (*,1505) pcoeff(1,1),pcoeff(2,2)
      xref(3)=xref(3)+coeff(1)
      write (*,*) 'corrected semimajor axis is:',xref(3)
c
c     For giggles, what is n dot over two from the delta a
c     data only?
      ndoto2=-.75d+00*dsqrt(eterms(1))*xref(3)**-2.5d+00
     1      *coeff(2)
      write (*,*) 'ndot over two from this data is',ndoto2
```

```
c     Zero out the coeff to avoid screwing up the dynamics in
c     the next section:
      coeff(1)=0.0d+00
      coeff(2)=0.0d+00
c
c     Now for the Big Least Squares run!  Copy the corrected state so
c     we screw up our only copy:
      do 3511 j=1,6
         vtemp(j)=xref(j)
 3511 continue
      dt=-(t-tlast)
      call esat6(dt,vtemp,randv,phi,4,1)
c     Use equation 6-10 from prospectus to propagate the covariance
c     backwards.  (Note that we DON'T make use of the noise matrix in
c     this determanistic portion!):
      call convp(pplus,mtemp,phi,6,6)
c
c     Form estimates for the differences w and w/o drag.  (Be
c     careful w/ quadrant on mean longitude:  experiments w/ perfect
c     data shows that we expect less than .1 radian difference
c     between predicted (that w/o drag) and observed mean longitude
c     (that w/ drag) over the span of one week.
c
      stemp=oldxhat(4,34)-vtemp(4)
      if (dabs(stemp) .gt. 3.0d+00) then
c       Assume this is a quadrant problem:
        if (stemp .gt. 0) then
           stemp=stemp-6.283185307179586d+00
        else
           stemp=stemp+6.283185307179586d+00
        endif
      endif
      mldiff(2)=stemp
      mlsig2(2)=oldpplus(4,4,34)+mtemp(4,4)
      adiff(2)=oldxhat(3,34)-vtemp(3)
      asig2(2)=oldpplus(3,3,34)+mtemp(3,3)
      time(2)=dt
c     Now for the remaining stored data points (33 more):
      do 1106 i=33,1,-1
        dt=dt-deltat*ibetween
c       Since we screw up the calling element array, copy it to a
c       temp array before the call:
        do 1107 j=1,6
           vtemp(j)=xref(j)
 1107   continue
        call esat6(dt,vtemp,randv,phi,4,1)
c
c       Now propagate covariance:
        call convp(pplus,mtemp,phi,6,6)
c
c       Form estimates for the differences w and w/o drag.  (Be
c       careful w/ quadrant on mean longitude:  experiments w/ perfect
c       data shows that we expect less than .1 radian difference
c       between predicted (that w/o drag) and observed mean longitude
```

```
c       (that w/ drag) over the span of one week.
c
        stemp=oldxhat(4,i)-vtemp(4)
        if (dabs(stemp) .gt. 3.0d+00) then
c         Assume this is a quadrant problem:
          if (stemp .gt. 0) then
             stemp=stemp-6.283185307179586d+00
          else
             stemp=stemp+6.283185307179586d+00
          endif
        endif
        mldiff(36-i)=stemp
        mlsig2(36-i)=oldpplus(4,4,i)+mtemp(4,4)
        adiff(36-i)=oldxhat(3,i)-vtemp(3)
        asig2(36-i)=oldpplus(3,3,i)+mtemp(3,3)
        time(36-i)=dt
 1106 continue
c
c
c     Now we form the two sums required for the weighted least
c     squares routine:
      do 1111 j=1,3
         do 1112 i=1,3
            sum1(i,j)=0.0d+00
 1112    continue
            sum2(j)=0.0d+00
 1111 continue
      write (*,*)
c     First the terms from delta semimajor axis:
      stemp=-2.0d+00/3.0d+00/dsqrt(eterms(1))*xref(3)**
     1    2.5d+00
      stemp2=stemp*stemp
      stmpt2=2.0d+00*stemp
      write (*,*) 'The data for delta semimajor axis is:'
      do 1113 i=1,35
c       We aren't using weights, so...
        asig2(i)=1.0d+00
        write (*,1501) time(i),adiff(i),asig2(i)
        dt=time(i)
        dt2=dt*dt
        sum1(2,2)=sum1(2,2)+stemp2/asig2(i)
        sum1(2,3)=sum1(2,3)+stmpt2*stemp*dt/asig2(i)
        sum1(3,2)=sum1(3,2)+stmpt2*stemp*dt/asig2(i)
        sum1(3,3)=sum1(3,3)+stmpt2*stmpt2*dt2/asig2(i)
        sum2(2)=sum2(2)+adiff(i)*stemp/asig2(i)
        sum2(3)=sum2(3)+adiff(i)*stmpt2*dt/asig2(i)
 1113 continue
      write (*,*)
c     Now the terms from delta mean longitude:
      write (*,*) 'The data for delta mean longitude is:'
      do 1114 i=1,35
c        We aren't using weights....
         mlsig2(i)=1.0d+00
         write (*,1501) time(i),mldiff(i),mlsig2(i)
```

```
            dt=time(i)
            dt2=dt*dt
            dt3=dt2*dt
            dt4=dt2*dt2
            dt=dt/mlsig2(i)
            dt2=dt2/mlsig2(i)
            dt3=dt3/mlsig2(i)
            dt4=dt4/mlsig2(i)
            suml(1,1)=suml(1,1)+1.0d+00/mlsig2(i)
            suml(1,2)=suml(1,2)+dt
            suml(1,3)=suml(1,3)+dt2
            suml(2,1)=suml(2,1)+dt
            suml(2,2)=suml(2,2)+dt2
            suml(2,3)=suml(2,3)+dt3
            suml(3,1)=suml(3,1)+dt2
            suml(3,2)=suml(3,2)+dt3
            suml(3,3)=suml(3,3)+dt4
            sum2(1)=sum2(1)+mldiff(i)/mlsig2(i)
            sum2(2)=sum2(2)+mldiff(i)*dt
            sum2(3)=sum2(3)+mldiff(i)*dt2
 1114    continue
c       Get the covariance of the coefficents:
        do 12 i=1,3
           do 11 j=1,3
              pcoeff(i,j)=0.0d+00
   11      continue
           pcoeff(i,i)=1.0d+00
   12   continue
        call ludcmp(suml,3,3,indx,stemp)
        do 13 j=1,3
           call lubksb(suml,3,3,indx,pcoeff(1,j))
   13   continue
c       Now we need to multiply pcoeff times sum2
        do 1115 i=1,3
           coeff(i)=0.0d+00
           do 1116 j=1,3
              coeff(i)=coeff(i)+pcoeff(i,j)*sum2(j)
 1116      continue
 1115   continue
        write (*,*)
        write (*,*) 'the coeff are '
        write (*,1505) coeff
        write (*,*)
        write (*,*) 'with a covariance of '
        write (*,1505) pcoeff
        write (*,*)
        write (*,*) 'a corrected mean longitude at last time with'
        write (*,*) 'data is:'
        write (*,*) xref(4)+coeff(1)
c
c       We're going to propage forward from reftime.  Since the
c       numerical derivative used to get phi falls apart when
c       used over long times, we stop computing the covariance
c       estimates.  Instead, keep it constant at the last value.
```

```
c     Since a call to esat6 with k=5 screws up the calling elements,
c     always copy our reference elements and use the copy in
c     the call.
c     Let's convert the last stored covariance to prvh for output.
c     (Note that the last stored covariance is p(-) for the first
c     predicted time!  An interesting extra....)
      call formp(u,d,pminus)
      call esat6(0.0d+00,xhat,randv,capr,1,1)
      call convp(pminus,prandv,capr,6,6)
c     call rot2rvh(randv,prandv,rmag,vmag,hmag,prvh)
c
c     Now start the prediction loop.  Start by fixing the mean
c     longitude at epoch (last time w/ data).
      xref(4)=xref(4)+coeff(1)
      t=t+deltat
  140 if (t .gt. endtime) goto 9998
         dt=t-reftime
c        Now we propagate the state forward (again copying the
c        state to avoid screwing it up):
         do 153 i=1,n
            xhat(i)=xref(i)
  153    continue
         call esat6(dt,xhat,randv,phi,5,0)
c
c        No need to play with the covariance since it's being kept
c        constant at the last stored value.
c        write (2,1501) (pminus(jj,jj),jj=1,6)
         write (4,1501) (prandv(jj,jj),jj=1,6)
c        write (7,1501) (prvh(jj,jj),jj=1,3)
c        We did, however, lose our calculation of rmag, vmag,
c        and hmag by not having the call to rot2rvh to find these
c        values from randv, so we better do it now:
c        rmag=dsqrt(randv(1)*randv(1)+randv(2)*randv(2)+
c    1     randv(3)*randv(3))
c        vmag=dsqrt(randv(4)*randv(4)+randv(5)*randv(5)+
c    1     randv(6)*randv(6))
c        vtemp(1)=randv(2)*randv(6)-randv(3)*randv(5)
c        vtemp(2)=randv(3)*randv(4)-randv(1)*randv(6)
c        vtemp(3)=randv(1)*randv(5)-randv(2)*randv(4)
c        hmag=dsqrt(vtemp(1)*vtemp(1)+vtemp(2)*vtemp(2)+
c    1    vtemp(3)*vtemp(3))
c        write (9,1501) rmag,vmag,hmag
c        write (3,1501) xhat
         write (5,1501) randv
c        Done...move on to next time...
         t=t+deltat
         goto 140
 9998 continue
c     We're done with this case, prediction and all..
      write (*,*) 'Predicted Refel at final time would be:'
      write (*,1501) xhat
      write (*,*)
      write (*,*) 'P(-) for refel at this time would be:'
      do 936 i=1,6
```

```
            write (*,1501) (pminus(i,j),j=1,6)
  936 continue
      write (*,*) 'finished with case ',case
      write (*,*)
c     close(2)
c     close(3)
      close(4)
      close(5)
c     close(7)
      close(8)
c     close(9)
      close(10)
 9999 continue
      stop
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine Esat6(t,elemts,randv,phi,k,mode)
c
c     This is a driver for Kaare Aksnes' version of Dirk
c     Brouwer's first order theory.  It includes J2, J3
c     J4, J5, and has had the effect of drag on the Mean
c     Anomaly added to it.  This driver allows the effect
c     of drag on the Semimajor axis to be included also.
c
c     W. Wiesel, August 1985
c
c     Modified by Kerry Hicks, AFIT, May 1988, 5 August 1988
c                              5 January 1989, 11 January 1989
c
c     Significant changes by Hicks:
c         1) The following are the only VERIFIED working
c            combinations for calling ESAT6 (Aksnes theory
c            still supports it's normal stuff as long as the
c            drag parameters stored in the common block "fudge"
c            are set to zero.):
c               k=1, mode=0 (For t<>0, drag effects on semimajor
c                           axis are ignored.)
c               k=1, mode=1 (Note that drag effects on semimajor
c                           axis are ignored for this situation.)
c                           (This doesn't matter for my problem, as
c                           this combination is only used in the
c                           stochastic "where am I" problem where
c                           propagation is over short times or I'm
c                           using the call to get the rotation matrix
c                           "capital R.")
c               k=2, mode=0 (I don't k=2, so it has not been verified!
c                           but will ignore the drag effects on a if
c                           t<>0)
c               k=2, mode=1 (ditto the above)
c               k=3, mode=0
```

```
c               k=3, mode=1 (illegal combination)
c               k=4, mode=0 (for t<>0, ignores drag variation on a)
c               k=4, mode=1 (ditto)
c                           (Note that k=4 is intended for the
c                           stochastic portion where I ignore
c                           drag.)
c               k=5, mode=0
c               k=5, mode=1
c                           (Note that k=5 is slower than k=4 so
c                           it should only be used when propagating
c                           over times when drag effects are signifi-
c                           cant.)
c
c         2) Drag effects are stored in the common block "fudge."
c            The linear and quadratic corrections to mean anomoly
c            take place within Aksnes' theory (subroutine aksnes) and
c            the corrections to the semimajor axis takes place in
c            this routine.  Mean longitude at epoch corrected in
c            main routine
c
c         3) This driver uses the equinoctal elements given below.
c            These elements were chosen because they are better defined
c            than the classical set when i or e is near zero.  Since
c            the estimation problem iterates, it's better to use well
c            defined variables!
c
c     VARIABLES:
c     t = time
c     elemts: These are the equinoctal elements:  af, ag,
c             semimajor axis (LUs), mean longitude (radians),
c             chi, and psi.
c     randv: x,y,z,x dot,y dot, z dot
c
c     coeff:  drag factors, as described above.
c             1 - delta mean longitude at epoch (used in main)
c             2 - delta n at epoch (delt -- sub-zero)
c             3 - n-dot-over-2
c
c     OPTIONS:
c     mode=0, calc orbit only
c     mode = 1, partials matrix too
c
c     k = 1  initialize theory & calc r and v
c            (If mode=1, then phi is really partial randv/partial
c            elements.)
c     k = 3  calc mean elements from input r and v (not verified with
c            drag terms....I never needed it)
c     k = 4  update elemts to new epoch time t
c            (If mode=1, then phi is partial(new elemts)/partial(old
c            elemts.)  (Drag effects on semimajor axis are ignored.)
c     k = 5  update elemts to new epoch time t.
c            (If mode=1, then phi is partial(new elemts)/partial (old
c            elemts.)  (Drag effects ARE INCLUDED.)
c
```

```
      double precision t,elemts(6),randv(6),phi(6,6),const(24)
     1   ,coeff(3),gamma
      common /earth/ re,b(5)
      common /units/ du,tu,vu
      common /fudge/ coeff
      double precision du,tu,vu
      double precision re,b
      double precision delemt(36),dconst(144),drandv(6)
      double precision newele(6),del(6)
c
c     displacement for numerical partials
c
      do 1000 i=1,6
         del(i)=elemts(i)*1.0d-06
 1000 continue
      if (k .eq. 4) go to 200
      if (k .eq. 5) go to 500
c
c     initialize theory, calc r and v, or get elemtnts from r and v
c
      call ak6drv(k,t,elemts,randv,const)
      if(k .eq. 3) then
c       Correct the semimajor axis at epoch (subtract correction
c       because we are "backing up" to epoch):
        elemts(3)=elemts(3) + 4.0d+00/3.0d+00/dsqrt(b(1))*
     2      elemts(3)**2.5d+00*coeff(3)*t
        return
      endif
      if (mode .eq. 0) return
      if(k .eq. 2) go to 150
c
c     initialize displaced orbits for partials
c
      do 120 i = 1,6
      do 119 j = 1,6
  119 delemt(6*(i-1)+j) = elemts(j)
  120 delemt(7*i-6) = delemt(7*i-6) + del(i)
c
  150 continue
c
c     calculate numerical partials matrix
c
      do 180 i = 1,6
      call ak6drv(k,t,delemt(6*(i-1)+1),drandv,dconst(24*(i-1)+1) )
      do 180 j = 1,6
  180 phi(j,i) = ( drandv(j) - randv(j) )/del(i)
      return
c
c
  200 continue
c
c     reinitialize epoch time while ignoring the effect of drag
c     on semimajor axis.
c
```

```
      call ak6drv(1,t,elemts,randv,const)
      call ak6drv(3,0.d0,newele,randv,const)
c     done?
      if(mode .eq. 0) go to 300
c
c     obtain partials matrix of reinitialization
c
      do 220 i = 1,6
         do 219 j = 1,6
            delemt(6*(i-1)+j) = elemts(j)
  219     continue
      delemt(7*i-6) = delemt(7*i-6) + del(i)
  220     continue
c         partials matrix
      do 280 i=1,6
         call ak6drv(1,t,delemt(6*(i-1)+1),drandv,dconst(1))
         call ak6drv(3,0.d0,delemt(6*(i-1)+1),drandv,dconst(1))
         do 281 j = 1,6
            phi(j,i) = ( delemt(6*(i-1)+j) - newele(j))/del(i)
  281    continue
  280 continue
      goto 300
c
  500 continue
c
c     reinitialize epoch time while including effect of drag on
c     semimajor axis.
c
      call ak6drv(1,t,elemts,randv,const)
      call ak6drv(3,0.d0,newele,randv,const)
c     Aksnes' theory took the semimajor axis forward as a constant.
c     Use the drag effect to fix this:
      gamma=-2.0d+00/3.0d+00/dsqrt(b(1))*elemts(3)**2.5d+00
      newele(3)=elemts(3) + gamma*coeff(2) + 2.0d+00*gamma*coeff(3)*t
c     Now get the correct randv for this:
      call ak6drv(1,0.0d+00,newele,randv,const)

c     done?
      if(mode .eq. 0) go to 300
c
c     obtain partials matrix of reinitialization
c
      do 520 i = 1,6
         do 519 j = 1,6
            delemt(6*(i-1)+j) = elemts(j)
  519     continue
      delemt(7*i-6) = delemt(7*i-6) + del(i)
  520     continue
c         partials matrix
      do 580 i=1,6
         call ak6drv(1,t,delemt(6*(i-1)+1),drandv,dconst(1))
         call ak6drv(3,0.d0,delemt(6*(i-1)+1),drandv,dconst(1))
         delemt(6*(i-1)+3)=delemt(6*(i-1)+3) - 4.0d+00/3.0d+00/
     1   dsqrt(b(1))*delemt(6*(i-1)+3)**2.5d+00*coeff(3)*t
```

```
      call ak6drv(1,0.0d+00,delemt(6
      do 581 j = 1,6
         phi(j,i) = ( delemt(6*(i-1)
581   continue
580 continue

300 continue
    do 310 i = 1,6
310 elemts(i) = newele(i)
    return
    end

------------------------------------

    subroutine ak6drv(k,t,fg,randv,co

    This routine drives the Aksnes th
    elements to Aksnes' reference ele
    code and converts his elements to

    double precision t,fg(6),ak(6),ra
   1   longp,tanio2,sra,arctan

    if (k .ne. 3) then
       Convert FG to AK:
       ak(6)=arctan(fg(5),fg(6))
       longp=arctan(fg(2),fg(1))
       ak(5)=longp-ak(6)
       if (ak(5) .lt. 0.0d+00) ak(5)=
       ak(3)=fg(4)-longp
       if (ak(3) .lt. 0.0d+00) ak(3)=
       sra=dsin(ak(6))
       ak(4)=2.0d+00*arctan(fg(5),sra)
       if (ak(4) .gt. 6.2831853071795
   1      -6.283185307179586d+00
       ak(2)=dsqrt(fg(1)*fg(1)+fg(2)*
       ak(1)=fg(3)
    endif

    call aksnes(k,t,ak,randv,const)

    if (k .eq. 3) then
      Convert AK to FG
      longp=ak(5)+ak(6)
      if (longp .gt. 6.283185307179586
   1     -6.283185307179586d+00
      tanio2=dtan(ak(4)/2.0d+00)
      fg(1)=ak(2)*dcos(longp)
      fg(2)=ak(2)*dsin(longp)
      fg(3)=ak(1)
      fg(4)=longp+ak(3)
      if (fg(4) .gt. 6.283185307179586
```

```
     1     -6.283185307179586d+00
           fg(5)=tanio2*dsin(ak(6))
           fg(6)=tanio2*dcos(ak(6))
         endif
c
         return
         end
c
c
c-----------------------------------------------------------------------
c
c
         subroutine terra
c
         common /earth/ re,b(5)
         common /units/ du,tu,vu
         double precision du,tu,vu
         double precision re,b
c
c        initialize data for earth: radius(DU),GM(DU**3/TU**2),J2,J3,J4
c        Values from JPL DE118
c
         re = 1.d0
         b(1) =  1.d0
         b(2) =  .10826270d-02
         b(3) =  -.25364140d-05
         b(4) =  -.16233497d-05
         b(5) =  -.22608567d-06
c
c        conversion constants: 1 DU = km, 1 TU sec, 1 VU in km/sec
         du =  6378.14d+00
         tu =  806.81168475130d+00
         vu =  7.90536394115830d+00
c        note: these definitions can be overridden by reinitializing
c        the commons /earth/ and /units/ in a main program
c        this may endanger the scaling used to obtain numerical partials
c
         return
         end
c
c
c-----------------------------------------------------------------------
c
c
         subroutine aksnes(k,t,oe,pos,const)
c
c        calculates position and velocity from mean elements
c        and vice versa
c
c        Dirk Brouwer's first order satellite theory as modified
c        by Kaare Asknes- perturbations in terms of the Hill variables
c        Programmed by K. Asknes, JPL, 1970.
c        Modified by W. Wiesel, AFIT, 1985
c        Modified by Kerry Hicks, AFIT, May 1988, 5 August 1988
```

```
c
c     k = 1 for initialization and computation of pos. and vel
c     k = 2 for computation of pos. and vel. without initialization
c           (a previous call with k=1 must have occured)
c     k = 3 for computation of mean elements from pos. and vel.
c
c     t = time
c     oe(1) = semi-major axis
c     oe(2) = eccentricity
c     oe(3) = mean anomaly in radians at epoch (t=0)
c     oe(4) = inclination in radians
c     oe(5) = argument of perigee in radians at epoch
c     oe(6) = right ascension of ascending node, radians, at epoch
c     re = equitorial radius of primary
c     b(1) = GM (unit of length**3/unit of time**2)
c     b(2),b(3),b(4),b(5) = geopotential constants J2, J3, J4, J5
c     pos = x,y,z,xdot,ydot,zdot
c
      dimension oe(6),pos(6),hv(7),hvo(7),err(7),sc(7)
      double precision t,oe,re,b,pos,pi,tpi,c,s,go,ho,slo,sgo,sho
     1,hv,fi,su,cu,si,gamma,hvo,err,rd,r,u,sh,sf,cf,f,ql,q2,sl,c2,c4,e
     2,e2,eta2,eta,qo,ga,ga2,ga3,ga4,xn,sll,sgl,shl,d,dl,sc,q3,sg,
     3 sine,cose,slml,clml,slpl,clpl,s2u,c2u,s2m2,c2m2,s2ml,c2ml,s2pl
     4,c2pl,drd,dr,du,dh,dg,g,ci,st,ct,sfi,cfi,arctan
      double precision const(24),theory(24),coeff(3)
      common /fudge/ coeff
      common /earth/ re,b(5)
c
c     SPECIAL NOTE:  Because this routine relies on values being
c     kept around after an initialization call (k=1), some fortrans
c     will require you to add the "SAVE" command to this routine.  The
c     version on AFIT'S ICC minisupercomputer and Microsoft Fortran
c     Version 3.2 for MSDOS do NOT need this command.  Note that the
c     inclusion of the SAVE command will NOT harm the execution if
c     it is not required....it's just an extra precaution you might
c     want to take if you're not sure.....It is not included here
c     because, while I don't know for sure, it might slow the execution
c     a bit.
c
      t2=t*t
      pi = 3.141592653589793d0
      tpi = 2.d0*pi
      go to (5,50,10),k
c
c     initialization
c
    5 continue
      c = dcos(oe(4))
      s = dsin(oe(4))
      go = dsqrt(b(1)*oe(1)*(1.d0-oe(2)*oe(2)))
      ho = go*c
      slo = oe(3)
      sgo = oe(5)
      sho = oe(6)
```

```
      go to 25
c     mean elements from position and velocity
c     compute instantaneous Hill variables hv
   10 iter = -1
      hv(4) = dsqrt(pos(1)*pos(1) + pos(2)*pos(2) + pos(3)*pos(3))
      hv(1) = (pos(1)*pos(4) + pos(2)*pos(5) + pos(3)*pos(6))/hv(4)
      hv(2) = dsqrt( dabs(pos(2)*pos(6)-pos(3)*pos(5))**2.d0
     1              +dabs(pos(3)*pos(4)-pos(1)*pos(6))**2.d0
     2              +dabs(pos(1)*pos(5)-pos(2)*pos(4))**2.d0 )
      hv(3) = pos(1)*pos(5)-pos(2)*pos(4)
      c = hv(3)/hv(2)
      fi = arctan(pos(2),pos(1))
      su = pos(3)/hv(4)
      cu = (hv(4)*pos(6) - hv(1)*pos(3))/hv(2)
      si = dsqrt(su*su+cu*cu)
      gamma = b(2)*( b(1)*re )**2.d0 / dabs(hv(2))**4.d0
      if( si .lt. gamma ) go to 13
      hv(5) = arctan(su,cu)
      hv(6) = fi - arctan(c*su,cu)
      if(hv(6) .lt. 0.d0 ) hv(6) = hv(6) + tpi
      hv(7) = 0.d0
      go to 14
   13 hv(5) = su
      hv(6) = cu
      hv(7) = fi
   14 do 15 i = 1,7
      hvo(i) = hv(i)
   15 err(i) = 0.d0
c     calculate mean Hill variables hvo by iteration
   20 rd = hvo(1)
      go = hvo(2)
      ho = hvo(3)
      r = hvo(4)
      u = hvo(5)
      sh = hvo(6)
      sf = go*rd/b(1)
      cf = go*go/b(1)/r - 1.d0
      f = arctan(sf,cf)
      oe(1) = b(1)/(-rd*rd + 2.d0*b(1)/r-dabs(go/r)**2.d0)
      oe(2) = dsqrt(sf*sf + cf*cf)
      q1 = r*rd/dsqrt(b(1)*oe(1))
      q2 = 1.d0 - r/oe(1)
      sl = arctan(q1,q2) - q1
      if(sl .lt. 0.d0 ) sl = sl + tpi
      c = ho/go
      s = dsqrt(dabs((1.d0+c)*(1.d0-c)))
      if(si .ge. gamma) go to 25
      u = arctan(hvo(5),hvo(6))
      sh = hvo(7) - u
      s = dsqrt(hvo(5)*hvo(5) + hvo(6)*hvo(6))
   25 c2 = c*c
      c4 = c2*c2
      e = oe(2)
      e2 = e*e
```

```
      eta2 = 1.d0 - e2
      eta = dsqrt(eta2)
      qo = oe(1)*(1.d0 - e2)
      ga = b(2)*dabs(re/qo)**2.d0
      ga2 = ga*ga
      ga3 = 0.d0
      ga4 = 0.d0
      if(b(2) .eq. 0.d0) go to 26
      ga3 = b(3)/b(2)**2.d0 * qo/re
      ga4 = b(4)/b(2)**2d0
   26 xn = dsqrt(b(1)/oe(1)**3.d0)
      sl1 = -0.75d0*eta*xn*ga*(1.d0-3.d0*c2-1.d0/32.d0*ga*(
     1     -15.d0+16.d0*eta+25.d0*eta2 + ( 30.d0-96.d0*eta-90.d0*
     2     eta2)*c2 + (105.d0+144.d0*eta+25.d0*eta2)*c4 - 15.d0*ga4*(
     3     3.d0-30.d0*c2+35.d0*c4)*e2))
      sgl = -0.75d0*xn*ga*(1.d0-5.d0*c2 -1.d0/32.d0*ga*(
     1     -35.d0+24.d0*eta+25.d0*eta2 + (90.d0-192.d0*eta-126.d0*
     2     eta2)*c2 + (385.d0+360.d0*eta+45.d0*eta2)*c4 - 5.d0*
     3     ga4*(4.d0*(3.d0-36.d0*c2+49.d0*c4) + 9.d0*(1.d0-14.d0*c2
     4     +21.d0*c4)*e2)))
      sh1 = -1.5d0*xn*ga*c*(1.d0+1.d0/16.d0*ga*(5.d0-12.d0*eta-9.d0*
     1     eta2 + (35.d0+36.d0*eta+5.d0*eta2)*c2 + 5.d0*ga4*(3.d0-
     2     7.d0*c2)*(2.d0+3.d0*eta)))
      d = (1.d0-c2)*(1.d0-15.d0*c2+5.d0*ga4*(1.d0-7.d0*c2))/(1.d0
     1     -5.d0*c2)
      d1 = (-11.d0+30.d0*c2-75.d0*c4-5.d0*ga4*(3.d0-14.d0*c2+35.d0*c4))
     1     /dabs(1.d0-5.d0*c2)**2.d0
c     save theory constants
      do 27 ii = 1,24
   27 const(ii) = theory(ii)
      if(k.ne.3) go to 50
      iter = iter + 1
c     compute the scaling factors of the Hill variables
      sc(1) = xn*oe(1)
      sc(2) = sc(1)*oe(1)
      sc(3) = sc(2)
      sc(4) = oe(1)
      sc(5) = tpi
      sc(6) = tpi
      sc(7) = tpi
      q3 = 0.d0
      do 30 i = 1,7
      q3 = dmax1(q3,dabs(hvo(i)-err(i))/sc(i))
   30 err(i) = hvo(i)
      if((iter.lt. 10) .and. (q3 .gt. 1.d-8)) go to 60
      if(iter .eq. 10) write(*,32) q3
   32 format(1x,'warning: error after 10 iterations =',d15.5)
      oe(4) = arctan(s,c)
      oe(3) = sl - t*(xn+sl1) - coeff(2)*t - coeff(3)*t2
      oe(5) = u - f - t*sgl
      oe(6) = sh - t*sh1
      do 35 i = 3,6
      j = oe(i)/tpi
      if(oe(i) .lt. 0.d0) j = j - 1
```

```
   35 oe(i) = (oe(i)-dble(j)*tpi)
      return
   50 continue
c
c     compute position and velocity at time t
c
      sl = t*(xn+sll) + slo + coeff(2)*t + coeff(3)*t2
      sg = t*sgl + sgo
      sh = t*shl + sho
      ql = idint(sl/tpi)
      if(sl .lt. 0.d0) ql = ql - 1.d0
      ql = tpi*ql
      sl = sl - ql
      call kepler(sl,e,sine,cose)
      r = oe(1)*(1.d0-e*cose)
      sf = eta*sine*oe(1)/r
      cf = (cose-e)*oe(1)/r
      rd = b(1)*e*sf/go
      f = arctan(sf,cf)
      u = sg+f+ql
      irev = u/tpi
      u = u - tpi*dble(irev)
      sf = e*sf
      cf = e*cf
   60 continue
      su = dsin(u)
      cu = dcos(u)
      slml = dsin(u-f)*e
      clml = dcos(u-f)*e
      slpl = dsin(u+f)*e
      clpl = dcos(u+f)*e
      q2 = 2.d0*u
      s2u = dsin(q2)
      c2u = dcos(q2)
      q3 = q2 - 2.d0*f
      s2m2 = e2*dsin(q3)
      c2m2 = e2*dcos(q3)
      q3 = q2 - f
      s2ml = e*dsin(q3)
      c2ml = e*dcos(q3)
      q3 = q2 + f
      s2pl = e*dsin(q3)
      c2pl = e*dcos(q3)
      q3 = 1.d0/(1.d0+eta)
      drd = -ga*qo*go/(2.d0*r*r)*((1.d0-c2)*s2u - d/8.d0*s2ml
     1      -ga3*cu*s + 0.5d0*(1.d0-3.d0*c2)*(q3+eta*dabs(r/qo)**2.d0
     2      )*sf)
      dr = ga*qo/4.d0*(     (1.d0-c2)*c2u - d/4.d0*c2ml + 2.d0*ga3*
     1      su*s + (1.d0-3.d0*c2)*(1.d0+q3*cf + 2.d0*eta*r/qo))
      du = -ga/4.d0*(      dl/4.d0*c2*s2m2 - (2.d0-5.d0*c2+0.5d0*d)*
     1      s2ml -0.5d0*(1.d0-7.d0*c2+d/4.d0*e2)*s2u + c2*s2pl -
     2      ga3*s*( 4.d0*cu+c1pl) + 3.d0*(1.d0-5.d0*c2)*(f-sl) +
     3      2.d0*(1.d0-6.d0*c2)*sf + (1.d0-3.d0*c2)*q3*sf*(2.d0+cf))
      dh = -ga*c/4.d0*(6.d0*sf-3.d0*s2ml-s2pl-dl/4.d0*s2m2 + 6.d0*(
```

```
     1     f-sl) - 3.d0*s2u)
      dg = ga*go/4.d0*((1.d0-c2)*(3.d0*c2m1+c2p1) - d/4.d0*c2m2 +
     1     2.d0*ga3*slm1*s + 3.d0*(1.d0-c2)*c2u)
      if(s .lt. ga) go to 65
      du = du + ga/4.d0*ga3*(1.d0+c2)/s*clm1
      dh = dh - ga/2.d0*c*ga3/s*clm1
   65 if(k .ne. 3) go to 70
      hvo(1) = hv(1) - drd
      hvo(2) = hv(2) - dg
      hvo(4) = hv(4) -dr
      hvo(5) = hv(5) - du
      hvo(6) = hv(6) - dh
      if(si .ge. gamma) go to 20
      hvo(5) = hv(5) -0.5d0*ga*ga3*cf
      hvo(6) = hv(6) + 0.5d0*ga*ga3*sf
      hvo(7) = hv(7) - (du+dh)
      go to 20
   70 continue
      rd = rd + drd
      r = r + dr
      u = u + du
      sh = sh + dh
      g = go + dg
      ci = ho/g
      si = dsqrt(dabs((1.d0+ci)*(1.d0-ci)))
      if(s .ge. ga) go to 80
      st = s*su + 0.5d0*ga*ga3*cf
      fi = u+sh
      q2 = g*(s*cu - 0.5d0*ga*ga3*sf)
      go to 100
   80 continue
      cu = dcos(u)
      su = dsin(u)
      st = si*su
      fi = sh + datan2(ci*su,cu)
      q2 = g*si*cu
  100 continue
      ct = dsqrt(dabs((1.d0+st)*(1.d0-st)))
      sfi = dsin(fi)
      cfi = dcos(fi)
      pos(1) = r*ct*cfi
      pos(2) = r*ct*sfi
      pos(3) = r*st
      q3 = dabs(r*ct)**2.d0
      pos(4) = pos(1)*rd/r - (pos(1)*st*q2 + pos(2)*ho)/q3
      pos(5) = pos(2)*rd/r - (pos(2)*st*q2 - pos(1)*ho)/q3
      pos(6) = pos(3)*rd/r + q2/r
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
```

```
c
      subroutine kepler(sl,e,sine,cose)
c
c      solve kepler's equation
c
      double precision sl,tpi,e,sine,cose,xl,xlo,eo,el,diff
      tpi = 6.2831853071795860d0
      xl = dmod(sl,tpi)
      eo = xl
      i = 1
   10 sine = dsin(eo)
      cose = dcos(eo)
      xlo = eo - e*sine
      el = eo + (xl-xlo)/(1.d0-e*cose)
      diff = el - eo
      if(dabs(diff) .lt. 1.d-12) go to 30
      if(i .gt. 49) go to 20
      eo = el
      i = i + 1
      go to 10
   20 write (*,25) diff
   25 format(1x,'Keplers equation did not converge',d13.5)
   30 continue
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
c
      function arctan(x,y)
      double precision arctan,a,x,y
      a = datan2(x,y)
      if( a.lt. 0.d0) a = a + 6.2831853071795860d0
      arctan = a
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine linearh(elemts,randv,xs,caph,capr,h)
c
c     This routine returns the linearized observation relation,
c     capital H as well as the expected observation h.
c     As a byproduct, it finds the capital R matrix
c     needed to convert the covariance from that for the elements
c     to that for randv.  Also returns an updated randv vector
c     for the elemts in the call.
c
      double precision elemts(6),randv(6),xs(3),caph(6),r,r2,r32
      double precision dot,capr(6,6),temp(3),h
```

```
c
c We'll need the capr matrix to apply the chain rule..
      call esat6(0.0d+00,elemts,randv,capr,1,1)
c
      r2=randv(1)*randv(1)+randv(2)*randv(2)+randv(3)*randv(3)
      r=dsqrt(r2)
      r32=r2**1.5d+00
      dot=randv(1)*xs(1)+randv(2)*xs(2)+randv(3)*xs(3)
c The nonzero elements of partial h/partial randv are:
      temp(1)=(xs(1)*r2-randv(1)*dot)/r32
      temp(2)=(xs(2)*r2-randv(2)*dot)/r32
      temp(3)=(xs(3)*r2-randv(3)*dot)/r32
c Need to convert this to partial h/partial elements...use
c chain rule.  Note that we make use of fact that the last
c three elements of temp would be zero.
      do 101 j=1,6
         caph(j)=0.0d+00
         do 111 i=1,3
            caph(j)=caph(j)+temp(i)*capr(i,j)
  111    continue
  101 continue
c Now for the expected observation...
      h=(randv(1)*xs(1)+randv(2)*xs(2)+randv(3)*xs(3))/r
c Got it all....
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine convp(pin,pout,r,nrows,ncols)
c
c     This routine converts from one covariance to another using the
c     rotation matrix r
c
      double precision pin(ncols,ncols),pout(nrows,nrows)
      double precision r(nrows,ncols),mtemp(6,6)
c nrows and ncols are the number of rows/columns in R
c
c Form R times Pin:
      do 10 i=1,nrows
        do 20 j=1,ncols
          mtemp(i,j)=0.0d+00
          do 30 k=1,ncols
            mtemp(i,j)=mtemp(i,j)+r(i,k)*pin(k,j)
   30     continue
   20   continue
   10 continue
c Now multiply (R times Pin) times R transpose:
      do 40 i=1,nrows
        do 50 j=1,nrows
          pout(i,j)=0.0d+00
          do 60 k=1,ncols
```

```
              pout(i,j)=pout(i,j)+mtemp(i,k)*r(j,k)
   60       continue
   50     continue
   40   continue
c  Done!
        return
        end
c
c
c-----------------------------------------------------------------------
c
c
        subroutine formp(u,d,p)
c
c       This routine assembles P=UDUt
c
        double precision u(6,6),d(6),p(6,6),mtemp(6,6)
c
c  this is UD after taking advantage of the zeros in U:
        do 5 i=1,5
          mtemp(i,i)=d(i)
          do 10 j=i+1,6
             mtemp(i,j)=u(i,j)*d(j)
    10    continue
     5  continue
        mtemp(6,6)=d(6)
c  now UD times U transpose (after exploiting zeros in both):
        do 15 i=1,6
          do 20 j=1,6
            p(i,j)=0.0d+00
            do 25 k=j,6
              p(i,j)=p(i,j)+mtemp(i,k)*u(j,k)
    25      continue
    20    continue
    15  continue
        return
        end
c
c
c-----------------------------------------------------------------------
c
c
        subroutine rot2rvh(rv,pin,r,v,h,pout)
c
c       This routine changes P for randv to P for rvh.
c       Also returns the values of r,v,h to the main
c       for printing.  Takes advantage of all of the zeros
c       in the rotation matrix.
c
        double precision rv(6),rotm(3,6),r,v,hvec(3),h,pin(6,6),
       1   pout(3,3),mtemp(3,6)
        r=dsqrt(rv(1)*rv(1)+rv(2)*rv(2)+rv(3)*rv(3))
        v=dsqrt(rv(4)*rv(4)+rv(5)*rv(5)+rv(6)*rv(6))
        hvec(1)=rv(2)*rv(6)-rv(3)*rv(5)
```

```
      hvec(2)=rv(3)*rv(4)-rv(1)*rv(6)
      hvec(3)=rv(1)*rv(5)-rv(2)*rv(4)
      h=dsqrt(hvec(1)*hvec(1)+hvec(2)*hvec(2)+hvec(3)*hvec(3))
c Assemble the nonzero parts of the rotation matrix:
      rotm(1,1)=rv(1)/r
      rotm(1,2)=rv(2)/r
      rotm(1,3)=rv(3)/r
      rotm(2,4)=rv(4)/v
      rotm(2,5)=rv(5)/v
      rotm(2,6)=rv(6)/v
      rotm(3,1)=(-hvec(2)*rv(6)+hvec(3)*rv(5))/h
      rotm(3,2)= (hvec(1)*rv(6)-hvec(3)*rv(4))/h
      rotm(3,3)=(-hvec(1)*rv(5)+hvec(2)*rv(4))/h
      rotm(3,4)= (hvec(2)*rv(3)-hvec(3)*rv(2))/h
      rotm(3,5)=(-hvec(1)*rv(3)+hvec(3)*rv(1))/h
      rotm(3,6)= (hvec(1)*rv(2)-hvec(2)*rv(1))/h
c R times Pin:
      do 10 j=1,6
        mtemp(1,j)=rotm(1,1)*pin(1,j)+rotm(1,2)*pin(2,j)+
     1     rotm(1,3)*pin(3,j)
        mtemp(2,j)=rotm(2,4)*pin(4,j)+rotm(2,5)*pin(5,j)+
     1     rotm(2,6)*pin(6,j)
        mtemp(3,j)=0.0d+00
        do 20 i=1,6
          mtemp(3,j)=mtemp(3,j)+rotm(3,i)*pin(i,j)
   20   continue
   10 continue
c Now do (RPin) times R transpose:
      do 30 j=1,3
        pout(j,1)=mtemp(j,1)*rotm(1,1)+mtemp(j,2)*rotm(1,2)+
     1    mtemp(j,3)*rotm(1,3)
        pout(j,2)=mtemp(j,4)*rotm(2,4)+mtemp(j,5)*rotm(2,5)+
     2    mtemp(j,6)*rotm(2,6)
        pout(j,3)=0.0d+00
        do 40 i=1,6
          pout(j,3)=pout(j,3)+mtemp(j,i)*rotm(3,i)
   40   continue
   30 continue
c That's all, folks.....
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine propcov(y,dtilda,cn,phi,uplus,dplus,n,u,d)
c
c     This routine propagates the covariance from t+ to (t+deltat)-
c     in factored form (see Maybeck, page 397).
c
      double precision y(6,12),dtilda(12),cn(6,6),phi(6,6),
     +   uplus(6,6),dplus(6),u(6,6),d(6),avecs(12,6),cvecs(12,6),
     +   dvecs(12,6)
```

```
      integer i,j,k,n
c Construct the y matrix:
      do 132 i=1,n
         do 135 j=1,n
            y(i,j)=0.0d+00
            do 140 k=1,j
               y(i,j)=y(i,j)+phi(i,k)*uplus(k,j)
  140       continue
  135    continue
  132 continue
      do 145 i=1,n
         do 150 j=1,n
            y(i,j+n)=cn(i,j)
  150    continue
  145 continue
c
c Construct dtilda:
      do 160 j=1,n
         dtilda(j)=dplus(j)
  160 continue
c
c Start algorithm on page 397 of Maybeck:
c
c Partition off the a vectors
      do 162 k=n,1,-1
         do 164 j=1,2*n,1
            avecs(j,k)=y(k,j)
  164    continue
  162 continue
c Now the "meat" of the algorithm......
      do 166 k=n,2,-1
         do 168 j=1,2*n,1
            cvecs(j,k)=dtilda(j)*avecs(j,k)
  168    continue
         d(k)=0.0d+00
         do 170 i=1,2*n,1
            d(k)=d(k)+avecs(i,k)*cvecs(i,k)
  170    continue
         do 172 i=1,2*n,1
            dvecs(i,k)=cvecs(i,k)/d(k)
  172    continue
         do 174 j=1,k-1,1
            u(j,k)=0.0d+00
            do 176 i=1,2*n,1
               u(j,k)=u(j,k)+avecs(i,j)*dvecs(i,k)
  176       continue
            do 178 i=1,2*n,1
               avecs(i,j)=avecs(i,j)-u(j,k)*avecs(i,k)
  178       continue
  174    continue
  166 continue
      do 182 j=1,2*n,1
         cvecs(j,1)=dtilda(j)*avecs(j,1)
  182 continue
```

```
      d(1)=0.0d+00
      do 184 i=1,2*n,1
         d(1)=d(1)+avecs(i,1)*cvecs(i,1)
  184 continue
c Since u is upper unitary, put the ones on the diagonal....
      do 190 k=1,n
         u(k,k)=1.0d+00
  190 continue
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine ludcmp(a,n,np,indx,d)
c
c     This routine is from the book, "Numerical Recipes."
c
c     Given an N x N matrix A, with physical dimenstion NP, this
c     routine replaces it by the LU decomposition of a rowwise
c     permutation of itself.  A and N are input.  A is output,
c     arranged as shown in eq 2.3.14 of the book; INDX is an
c     output vector which records the row permutation effected by
c     the partial pivoting; D is output as +/-1 depending on whether
c     the number of row interchanges was even or odd, respectively.
c     This routine is used in combination with LUBKSB to solve linear
c     equations or to invert a matrx.
c
      implicit double precision (a-h,o-z)
      parameter (nmax=20,tiny=1.0e-20)
      dimension a(np,np),indx(n),vv(nmax)
      d=1.
      do 12 i=1,n
        aamax=0.
        do 11 j=1,n
          if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
        if (aamax.eq.0.) pause 'singular matrix.'
        vv(i)=1./aamax
12    continue
      do 19 j=1,n
        if (j.gt.1) then
          do 14 i=1,j-1
            sum=a(i,j)
            if (i.gt.1)then
              do 13 k=1,i-1
                sum=sum-a(i,k)*a(k,j)
13            continue
              a(i,j)=sum
            endif
14        continue
        endif
        aamax=0.
```

```
        do 16 i=j,n
          sum=a(i,j)
          if (j.gt.1)then
            do 15 k=1,j-1
              sum=sum-a(i,k)*a(k,j)
15          continue
            a(i,j)=sum
          endif
          dum=vv(i)*abs(sum)
          if (dum.ge.aamax) then
            imax=i
            aamax=dum
          endif
16      continue
        if (j.ne.imax)then
          do 17 k=1,n
            dum=a(imax,k)
            a(imax,k)=a(j,k)
            a(j,k)=dum
17        continue
          d=-d
          vv(imax)=vv(j)
        endif
        indx(j)=imax
        if(j.ne.n)then
          if(a(j,j).eq.0.)a(j,j)=tiny
          dum=1./a(j,j)
          do 18 i=j+1,n
            a(i,j)=a(i,j)*dum
18        continue
        endif
19    continue
      if(a(n,n).eq.0.)a(n,n)=tiny
      return
      end
c
c
c-----------------------------------------------------------------------
c
c
      subroutine lubksb(a,n,np,indx,b)
c
c     From the book, "Numerical Recipes."
c
c     Solves the set of N linear equations AX=B.  Here A is input,
c     not as the matrix A but rather as its LU decomposition,
c     determined by the routine LUDCMP.  INDX is input as the permu-
c     tion vector returned by LUDCMP.  B is input as the RHS vector B,
c     and returns with the solution vector X.  A, N, NP, and INDX
c     are not modified by this routine and can be left in place for
c     successive calls with different RHSs B.  This routine takes
c     into account the possibility that B will begin with many zero
c     elements, so it is efficient for use in matrix inversion
c
```

```
      implicit double precision (a-h,o-z)
      dimension a(np,np),indx(n),b(n)
      ii=0
      do 12 i=1,n
        ll=indx(i)
        sum=b(ll)
        b(ll)=b(i)
        if (ii.ne.0)then
          do 11 j=ii,i-1
            sum=sum-a(i,j)*b(j)
11        continue
        else if (sum.ne.0.) then
          ii=i
        endif
        b(i)=sum
12    continue
      do 14 i=n,1,-1
        sum=b(i)
        if(i.lt.n)then
          do 13 j=i+1,n
            sum=sum-a(i,j)*b(j)
13        continue
        endif
        b(i)=sum/a(i,i)
14    continue
      return
      end
```

Appendix C: Orbital Element Review

Six independent quantities are required to describe a spacecraft's position and velocity completely. Three coordinate systems are discussed in this study: a cartesian system (geocentric-equatorial) and two orbital element sets (classical and equinoctial). These systems are described briefly in this appendix.

The geocentric-equatorial system is a typical cartesian system, with three orthogonal axes $\hat{e}_x$, $\hat{e}_y$, $\hat{e}_z$ (7:55-56; 27:7-9). $\hat{e}_x$ points in the direction of the vernal equinox; $\hat{e}_z$ points out the Earth's north pole; $\hat{e}_y$ completes the system such that it lies in the Earth's equatorial plane and $\hat{e}_z = \hat{e}_x \times \hat{e}_y$, as shown in Figure C-1. In this system the position and velocity of the spacecraft are simply:

$$\underset{\sim}{r} = x\hat{e}_x + y\hat{e}_y + z\hat{e}_z \qquad \text{(C-1a)}$$

$$\underset{\sim}{v} = v_x\hat{e}_x + v_y\hat{e}_y + v_z\hat{e}_z \qquad \text{(C-1b)}$$

The six elements which completely describe the position and velocity are the scalar elements of the vectors $\underset{\sim}{r}$ and $\underset{\sim}{v}$.

The classical orbital elements are convenient for visualizing orbital motion (7:58-60; 27:42-44). These are defined with the help of Figure C-1 as:

1. a, the semimajor axis -- a constant defining the size of the conic section (orbit).

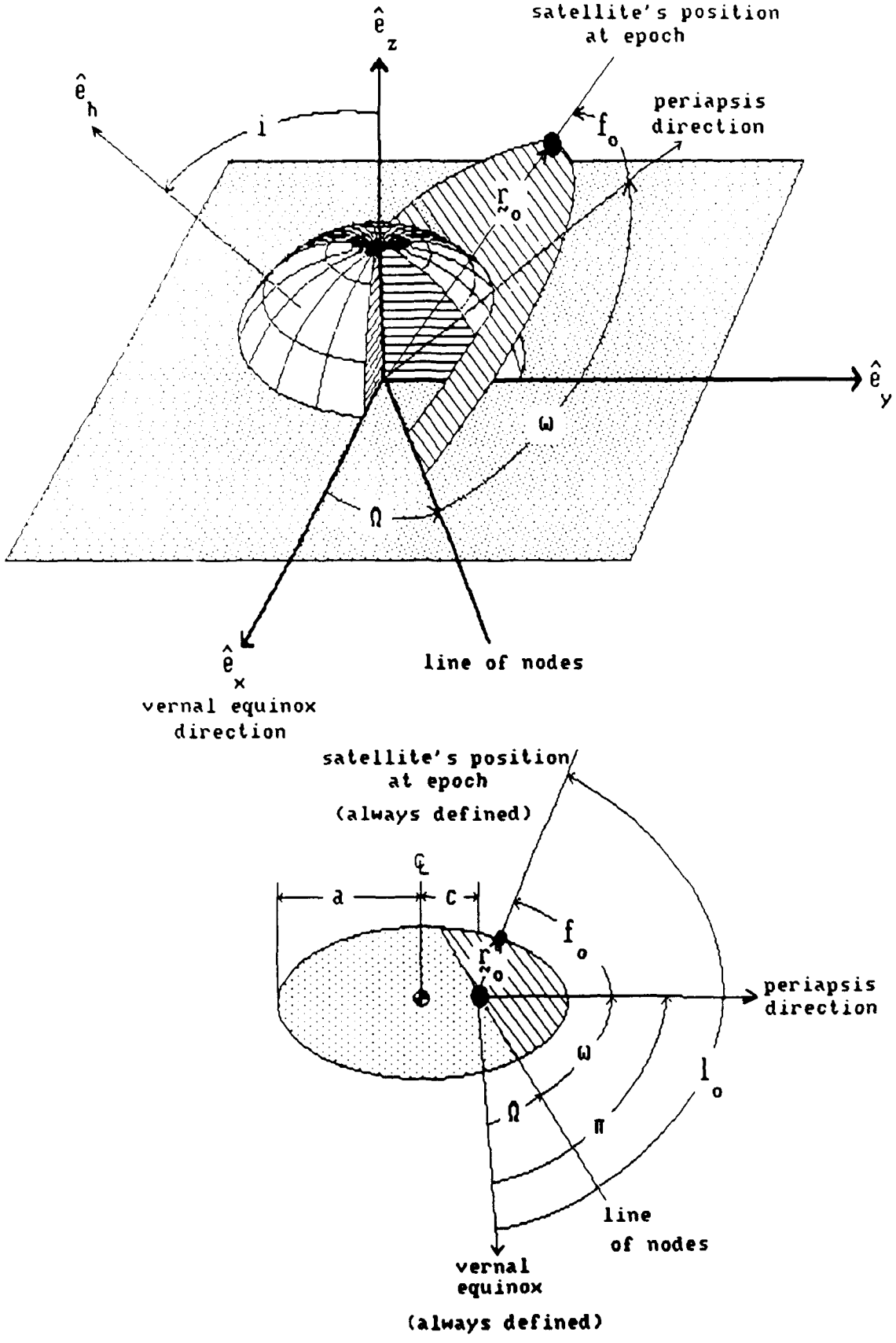

Figure C-1. Orbital Elements

2. e, eccentricity -- a constant defining the shape of the conic section (orbit). The value is given by $e = \frac{c}{a}$ where c is the distance shown in Figure C-1.

3. i, inclination -- the angle between $\hat{e}_z$ and the orbit normal $\hat{e}_h$.

4. Ω, longitude of the ascending node -- the angle (in the x-y plane) between $\hat{e}_x$ and the line of nodes, measured counterclockwise as viewed from above the $\hat{e}_z$ axis.

5. ω, argument of periapsis -- the angle (in the orbit plane) between the line of nodes and the periapsis direction, measured in the direction of satellite motion.

6. f_o, true anomaly at epoch -- the angle (in the orbit plane) between the periapsis direction and the current position vector, measured in the direction of satellite motion.

The true anomaly at epoch, f_o, is often replaced with the mean anomaly at epoch, M_o. At any time, the true anomaly and mean anomaly can be related by way of the eccentric anomaly E

$$M = E - e \sin E \tag{C-2}$$

where the eccentric anomaly is found by computing:

$$\cos E = \frac{\cos f + e}{1 + e \cos f} \tag{C-3a}$$

$$\sin E = \frac{\sqrt{1 - e^2}\ \sin f}{1 + e \cos f} \qquad \text{(C-3b)}$$

Note that the argument of periapsis is undefined for circular ($e = 0$) orbits and the longitude of the ascending node is undefined for equatorial ($i = 0^\circ$) orbits. Certain sums of these terms are still defined, however. The sum $\pi = \omega + \Omega$ is known as the longitude of the periapsis and exists even when Ω does not. Similarly, the true longitude at epoch, $l_o = \pi + f_o$, is defined for all orbits, regardless of whether ω or Ω are defined.

A set of equinoctial elements can be derived from the classical elements (8:490-495; 27:44; 28:3-8 - 3-9). These are

$$a_f = e \cos(\Omega + \omega) \qquad \text{(C-4a)}$$

$$a_g = e \sin(\Omega + \omega) \qquad \text{(C-4b)}$$

$$a = a \qquad \text{(C-4c)}$$

$$L = \Omega + \omega + M \qquad \text{(C-4d)}$$

$$\chi = \frac{\sin i \sin\Omega}{1 + \cos i} \qquad \text{(C-4e)}$$

$$\psi = \frac{\sin i \cos\Omega}{1 + \cos i} \qquad \text{(C-4f)}$$

where a is the semimajor axis; L is the mean longitude; e, i, Ω, M, and ω have already been defined; and a_f, a_g, χ, and ψ do not have formal names. Unlike the classical orbital elements, these are well-defined for all orbits studied.

Appendix D. Position and Velocity Terms for Freedom's Orbit

To obtain the position and velocity from Freedom's equinoctial elements, a simple algorithm can be followed (27:51,73-75). Only its application to circular orbits will be covered here. The full, non-circular case can be found in Reference 27. Definitions for the orbital elements employed herein are in Appendix C and will not be repeated.

First, compute r, $\dot{r}$, p, $r\dot{f}$, sinl, and cosl from the following sequence:

$$r = a(1 - e\cos E) \tag{D-1a}$$

$$\dot{r} = \frac{\sqrt{\mu a}}{r}(e \sin E) \tag{D-1b}$$

$$p = a\left(1 - e^2\right) \tag{D-1c}$$

$$r\dot{f} = \frac{\sqrt{\mu p}}{r} \tag{D-1d}$$

$$\text{sinl} = \frac{a}{r}\left[\sin(E + \pi) - a_g - a_f \frac{e \sin E}{1 + \sqrt{1 - e^2}}\right] \tag{D-1e}$$

$$\text{cosl} = \frac{a}{r}\left[\cos(E + \pi) - a_f + a_g \frac{e \sin E}{1 + \sqrt{1 - e^2}}\right] \tag{D-1f}$$

Which, for circular orbits, reduce to

$$r = a \tag{D-2a}$$

$$\dot{r} = 0 \tag{D-2b}$$

$$p = a \tag{D-2c}$$

$$r\dot{f} = \sqrt{\frac{\mu}{a}} = v_c \tag{D-2d}$$

$$\sin l = \sin L \tag{D-2e}$$

$$\cos l = \cos L \tag{D-2f}$$

where the term $\sqrt{\frac{\mu}{a}}$ has been noted as the circular orbit speed.

Define the terms w_x, w_y, w_z by the sequence of calculations:

$$w_z = \frac{1 - \psi^2 - \chi^2}{1 + \psi^2 + \chi^2} \tag{D-3a}$$

$$w_x = (1 + w_z)\,\chi \tag{D-3b}$$

$$w_y = -(1 + w_z)\,\psi \tag{D-3c}$$

Then, the position and velocity are given by

$$\underset{\sim}{r} = r\left(\hat{\underset{\sim}{f}}\cos l + \hat{\underset{\sim}{g}}\sin l\right) \tag{D-4a}$$

$$\underset{\sim}{v} = \dot{r}\left(\hat{\underset{\sim}{f}}\cos l + \hat{\underset{\sim}{g}}\sin l\right) + r\dot{f}\left(-\hat{\underset{\sim}{f}}\sin l + \hat{\underset{\sim}{g}}\cos l\right) \tag{D-4b}$$

where

$$\hat{\underline{f}} = \begin{bmatrix} 1 - \dfrac{w_x^2}{1 + w_z} \\ \dfrac{-w_x w_y}{1 + w_z} \\ -w_x \end{bmatrix} \qquad \text{(D-5a)}$$

$$\hat{\underline{g}} = \begin{bmatrix} \dfrac{-w_x w_y}{1 + w_z} \\ 1 - \dfrac{w_y^2}{1 + w_z} \\ -w_y \end{bmatrix} \qquad \text{(D-5b)}$$

Or, in light of Eqs. (D-2), Eqs. (D-4) become:

$$\underline{r} = a \left(\hat{\underline{f}} \cos L + \hat{\underline{g}} \sin L \right) \qquad \text{(D-6a)}$$

$$\underline{v} = v_c \left(-\hat{\underline{f}} \sin l + \hat{\underline{g}} \cos l \right) \qquad \text{(D-6b)}$$

For Freedom's orbit:

$$w_x \approx 0.3320 \qquad \text{(D-7a)}$$

$$w_y \approx -0.3320 \qquad \text{(D-7b)}$$

$$w_z \approx 0.8829 \qquad \text{(D-7c)}$$

Finally, the position and velocity vectors can be computed via Eqs. (D-5) and (D-6). These are simply:

$$\underset{\sim}{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \approx \begin{bmatrix} 0.9415\ \cos L + 0.0585\ \sin L \\ 0.0585\ \cos L + 0.9415\ \sin L \\ -0.3320\ \cos L + 0.3320\ \sin L \end{bmatrix} a \qquad \text{(D-8a)}$$

$$\underset{\sim}{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \approx \begin{bmatrix} 0.0585\ \cos L - 0.9415\ \sin L \\ 0.9415\ \cos L - 0.0585\ \sin L \\ 0.3320\ \cos L + 0.3320\ \sin L \end{bmatrix} v_c \qquad \text{(D-8b)}$$

Bibliography

1. Agrawal, Brij N. Design of Geosynchronous Spacecraft. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1986.

2. Aksnes, Kaare. "A Second-Order Artificial Satellite Theory Based on an Intermediate Orbit," The Astronomical Journal, 75: 1066-1076 (November 1970).

3. Aksnes, Kaare. A Second-Order Artificial Satellite Theory Based on an Intermediate Orbit. Contract NAS7-100. Pasadena, CA: Jet Propulsion Laboratory, California Institute of Technology, 1 November 1970 (Report No. 32-1507).

4. Aksnes, Kaare. "On the Use of the Hill Variables in Artificial Satellite Theory: Brouwer's Theory," Astronomy and Astrophysics, 17: 70-75 (January 1972).

5. Anthony, 1Lt John F. Design of an Orbital Element Estimator Using Relative Motion Data. MS Thesis, AFIT/GA/AA/81D-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1981.

6. Baker, Robert M., Jr. and Makemson, Maud W. An Introduction to Astrodynamics. New York: Academic Press, 1967.

7. Bate, Roger R. et al. Fundamentals of Astrodynamics. New York: Dover Publications, Inc., 1971.

8. Battin, Richard H. An Introduction to the Mathematics and Methods of Astrodynamics. New York: American Institute of Aeronautics and Astronautics, Inc., 1987.

9. Bierman, Gerald J. Factorization Methods for Discrete Sequential Estimation. New York: Academic Press, Inc., 1977.

10. Boden, Cpt Daryl G. Orbital Estimation Using Two-Body Classical Orbital Elements as Measurement Updates. MS Thesis, AFIT/GA/AA/79D-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1979.

11. Boden, Daryl G. A Comparison of Nonlinear Filters for Orbit Determination and Estimation. PhD Dissertation. University of Illinois, Urbana-Champaign IL, 1986 (AD-A170680).

12. Brouwer, Dirk. "Solution of the Problem of Artificial Satellite Theory Without Drag," The Astronomical Journal, 64: 378-397 (November 1959).

13. Carlson, Neal A. "Fast Triangular Formulation of the Square Root Filter," AIAA Journal, 11: 1259-1265: (September 1973).

14. Chory, M. A. et al. "Autonomous Navigation -- Where We Are in 1984," Guidance and Control Conference. Paper No. 84-1826. New York: American Institute of Aeronautics and Astronautics, 1984.

15. Deason, Billie, Public Relations Specialist. Telephone interview. Johnson Space Center, Houston, TX, 7 Oct 1988 and 7 Nov 1988.

16. Dyer, P. and McReynolds, S. "Extension of Square-Root Filtering to Include Process Noise," Journal of Optimization Theory and Applications, 3: 444-458 (June 1969).

17. Ferguson, Jackson. R., Jr. Autonomous Navigation of USAF Spacecraft. PhD Dissertation. University of Texas at Austin, Austin TX, 1983 (AD-A139173).

18. Fortmann, Thomas E. and Hitz, Konrad L. An Introduction to Linear Control Systems. New York: Marcel Dekker, Inc., 1977.

19. Gaposchkin, E. M. (Editor). 1973 Smithsonian Standard Earth (III). Cambridge: Smithsonian Institution Astrophysical Observatory, 28 Nov 1973 (SAO Special Report No. 353).

20. Gevarter, William B. and Heer, Ewald. "Requirements and Opportunities for Autonomous Systems in Space," Advances in the Astronautical Sciences, Vol 39: 127- 138. San Diego: Univelt, Inc., 1979.

21. Gounley, Robert et al. "Autonomous Satellite Navigation by Stellar Refraction," Journal of Guidance and Control, 7: 129-134 (March/April 1984).

22. Heuberger, Howard and Leonard Church. "Landsat-4/Global Positioning System Navigation Results," Advances in the Astronautical Sciences, Vol 54: 589-602. San Diego: Univelt, Inc., 1983.

23. Jazwinski, Andrew H. Stochastic Processes and Filtering Theory. New York: Academic Press, Inc., 1970.

24. Junkins, John L. An Introduction to Optimal Estimation of Dynamical Systems. Alphen Aan Den Rijn, The Netherlands: Sijthoff & Noordhoff International Publishers, 1978.

25. King-Hele, Desmond. Theory of Satellite Orbits in an Atmosphere. London: Butterworth, 1964.

26. Koelle, Heinz Hermann (Editor). Handbook of Astronautical Engineering. New York: McGraw-Hill Book Company, 1961.

27. Koskela, Paul E. Astrodynamical Analysis for the Advanced Orbit/Ephemeris Subsystem, DDI subcontract No. 66-1 to AF 04(695)-976. Newport Beach, CA: Philco-Ford Corporation, Aeronutronic Division, 1 Sep 1967 (Aeronutronic Publication No. U-4180).

28. Kwok, Johnny H. The Artificial Satellite Analysis Program (ASAP) Version 2.0 (User manual and software package). Pasadena, CA: The Jet Propulsion Laboratory, 20 April 1987 (EM 312/87-153).

29. Liu, A. S. "Autonomous Satellite Navigation Using the Stellar Horizon Atmospheric Dispersion Sensor," Advances in the Astronautical Sciences, Vol 54: 573-588. San Diego: Univelt, Inc., 1983.

30. Lowrie, D. W. "Autonomous Navigation Systems Technology Assessment," 17th Aerospace Sciences Meeting. Paper No. 79-0056. New York: American Institute of Aeronautics and Astronautics, January 1979.

31 Massey, Sir Harrie F.R.S. Space Physics. London: Cambridge University Press, 1964.

32. Maybeck, Peter S. Stochastic Models, Estimation, and Control: Volume 1. Orlando: Academic Press, Inc., 1979.

33. Maybeck, Peter S. Stochastic Models, Estimation, and Control: Volume 2. New York: Academic Press, Inc., 1982.

34. Mease, K. D. et al. "An Approach to Autonomous, Onboard Orbit Determination," AIAA/AAS Astrodynamics Conference. Paper No. 84-2031. New York: American Institute of Aeronautics and Astronautics, August 1984.

35. Metzler, Roger A. "Autonomous Satellite Navigation at Five Times Synchronous Altitude," Journal of Guidance and Control, 7: 62-68 (January/February 1984).

36. Press, William H. et al. Numerical Recipes. New York: Press Syndicate of the University Of Cambridge, 1986.

37. Roy, Archie E. Orbital Motion (Second Edition). Bristol: Adam Hilger, Ltd, 1982.

38. Schappel, R. T. "Advanced Guidance and Control Technology for Spacecraft Automation," Advances in the Astronautical Sciences, Vol 39: 139-154. San Diego: Univelt, Inc., 1979.

39. Thornton, Catherine L. and Bierman, Gerald J. A Numerical Comparison of Discrete Kalman Filtering Algorithms: An Orbit Determination Case Study: Technical Memorandum. Contract NAS 7-100. Pasadena, CA: Jet Propulsion Laboratory, 15 June 1976 (N76-26930).

40. Turner, Philip R. Autonomous Spacecraft Design and Validation Handbook (Issue I): Interim Report, Contract NAS 7-10, JPL Task Plan No. 80-1487, Rev. A. Pasadena CA: Jet Propulsion Laboratory, California Institute of Technology, 30 April 1982 (Report SD-TR-82-58).

41. United States Navel Observatory. The Astronomical Almanac for the Year 1985. Washington: U.S. Government Printing Office, 1984.

42. Ward, Cpt John E., Jr. Autonomous State Determination for an Earth Orbiting Satellite Using Horizon and Star Sensors. MS Thesis, AFIT/GA/AA/84D-12. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

43. Ward, Cpt Michael L. P. Estimated Satellite Cluster Elements in Near Circular Orbit. MS Thesis, AFIT/GA/AA/88D-13. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1988.

44. Wiesel, William E., Jr. Lecture material distributed in MECH 731, Modern Methods of Orbit Determination. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, January 1986.

45. Wiesel, William E., Jr., Associate Professor of Astronautics. Personal interviews. Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 21 Jan through 7 June 1988.

46. Wiesel William E., Jr., Associate Professor of Astronautics. Personal interview. Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 30 Jan 1989.

47. Zombeck, Martin V. Handbook of Space Astronomy and Astrophysics. Cambridge: Cambridge University Press, 1982.

Vita

Captain Kerry D. Hicks was born 6 October 1962 in Cordell, Oklahoma. He graduated from O'Fallon Township High School in O'Fallon, Illinois in 1981. The same year, he was admitted to the University of Illinois (Urbana-Champaign, Illinois). From there, he received a Bachelor of Science in Aeronautical and Astronautical Engineering with honors and was commissioned in the USAF through the ROTC program in May 1985. Captain Hicks also began the resident Master's Degree program at the School of Engineering, Air Force Institute of Technology (Wright-Patterson AFB, Ohio) in May of 1985. He was awarded his Master's Degree in Astronautical Engineering (Distinguished Graduate) in December 1986. In January 1987, he officially began his studies at AFIT towards his PhD under the Laboratory Scholars Program.

Permanent Address: 1512 Princeton Dr

O'Fallon, Il 62269

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/DS/AA/89-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Kerry D. Hicks, B.S., M.S., Captain, USAF	
13a. TYPE OF REPORT PhD Dissertation 13b. TIME COVERED FROM _____ TO _____ 14. DATE OF REPORT (Year, Month, Day) 1989, December 15. PAGE COUNT 226	
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Autonomous Navigation, Orbit Determination, Orbit Prediction, Estimation, Position Estimation
17	07	04	
22	03		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: AN AUTONOMOUS ORBIT DETERMINATION SYSTEM FOR EARTH SATELLITES

Research Committee Chairman: Dr. William E. Wiesel
Professor of Astronautical Engineering

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL William E. Wiesel, Prof. of Astro Eng	22b. TELEPHONE (Include Area Code) (513) 255-3517 22c. OFFICE SYMBOL AFIT/ENY

DD Form 1473, JUN 86 *Previous editions are obsolete.*

Block 19:

A new system for autonomous satellite navigation is developed and investigated. Unlike many previous studies, however, this system is not limited to the determination of an Earth satellite's current position. By using a two-step technique combined with a general perturbations model, this method allows the air-drag effect on the orbit to be estimated and applied to a future position prediction. Simulations using existing hardware have demonstrated that the algorithm presented is capable of current position estimates of sub-kilometer accuracy. Prediction precisions rival those of ground-based facilities: 8 - 12 kilometers two weeks in the future for low-Earth orbit.